VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE POZADÍ Z NĚKOLIKA FOTOGRAFIÍ

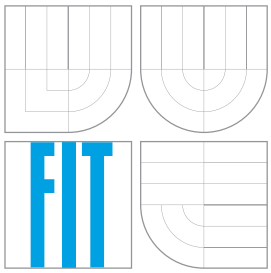
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR MOTÁČEK

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE POZADÍ Z NĚKOLIKA FOTOGRAFIÍ

BACKGROUND RECONSTRUCTION FROM SEVERAL PHOTOGRAPHS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR MOTÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL SEEMAN

BRNO 2010

Abstrakt

Tato práce se zabývá rekonstrukcí pozadí z několika fotografií (efekt tzv. vylidnění scény). Jsou zde představeny metody na získávání pozadí z videa a diskuze jejich využití pro práci s fotografiemi. Největší důraz je kladen na přístup modelování pozadí za pomocí směsi Gaussových funkcí (mixture of Gaussian) a s tím spjatá snaha o vylepšení tohoto algoritmu vzhledem ke statickému obrazu. U pořízených snímků se předpokládá scéna snímána ze stativu.

Abstract

This thesis concerns the background reconstruction from several photographs (so called depopulation scene efect). There are presented methods for obtaining the background from video and discussion of their use for photographs. The greatest emphasis is placed on the Gaussian mixture model and effort to improve this algorithm due to static image. The photographs should be taken with a tripod.

Klíčová slova

rekonstrukce pozadí, odhadnutí pozadí, segmentace pozadí, model pozadí, inicializace pozadí, statický obraz, OpenCv

Keywords

background reconstruction, background estimation, background segmentation, background model, background inicialization, still image, OpenCv

Citace

Vladimír Motáček: Rekonstrukce pozadí z několika
fotografií, diplomová práce, Brno, FIT VUT v Brně, 2010

Rekonstrukce pozadí z několika fotografií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Seemana.

.....
Vladimír Motáček
22. května 2010

Poděkování

Rád bych poděkoval svému vedoucímu diplomové práce Ing. Michalovi Seemanovi za drahocenný čas, vstřícnost a pomoc s pořízením testovacích dat.

Obsah

Kapitola 1

Úvod

Rekonstrukcí pozadí se myslí odstranění pohybujících se objektů, které nejsou pevnou součástí snímané scény. Pozadí potřebujeme rekonstruovat v řadě případů. Rekonstrukci můžeme rozdělit na dvě základní větve podle vstupních dat (rekonstrukci z videa nebo z fotografií.) Nejčastějším případem nasazení rekonstrukce pozadí je rozlišení pozadí od popředí ve videu. S masivním nástupem různých dohledových systémů právě detekce pozadí bývá kritická část celého výpočtu. Jakmile už máme k dispozici model pozadí, jsme schopni rozlišit objekty na popředí (automobily, chování chodců ve videu, apod.).

Druhou možností, jak již bylo zmíněno výše, je práce s fotografiemi. Její využití plyne z povahy rozdílu vůči videu. Největším rozdílem je rozlišení jednotlivých snímků u videa a fotografií. Pokud například požadujeme kvalitní obraz nějaké architektonické památky a nechceme mít v záběru různé rušivé elementy jako projíždějící auta nebo procházející osoby, jsou fotografie jedinou volbou. I kdyby byla scéna nasnímána HD videem, její rozlišení je buď 1280 x 720, nebo 1920 x 1080 pixelů. Využitím může být i sám umělecký dojem, například liduprázdného náměstí za bílého dne.

Diplomová práce si klade za cíl prozkoumat dosud známé metody získávání pozadí z fotografií a implementovat jednu z nich. U fotografií se předpokládá, že si jednotlivé pixely vzájemně odpovídají, a proto je vhodné, aby byly pořízeny ze stativu. Ukázalo se, že i přes veškerou snahu nejsem schopen najít žádnou literaturu pojednávající přímo o práci s fotografiemi. Naštěstí situace ve videu je mnohem uspokojivější a existuje celá řada materiálů zabývající se touto problematikou. Proto jsem se zpočátku zaměřil na extrakci pozadí (popředí) z videa. Na druhou stranu existuje oproti videu omezení spočívající v množství vstupních dat. Za úspěch se dá považovat, pokud bude mít algoritmus k dispozici něco kolem třiceti snímků a s tímto faktem se bude muset nějak vypořádat. Pro video je to zanedbatelný počet (asi 1 sekunda záznamu). Vzhledem k uvedeným faktům se vypsané téma diplomové práce lehce dotýká neprozkoumané nebo alespoň řádně nezdokumentované oblasti počítačového vidění, a proto bylo snahou přijít buď se zcela novým originálním řešením, nebo se přinejmenším snažit vybrat nejlepší dosud známý postup z videa a ten upravit pro svou potřebu, případně jakkoli vylepšit.

Tento dokument se skládá včetně úvodu z pěti kapitol. V druhé kapitole najdete teoretický základ k pochopení problému. Protože se jedná o rekonstrukci pozadí z barevných fotografií, důležité zastoupení zde sehrává role barevných prostorů a vzdálenostních funkcí. Čtenář se dozví, jak se dělá rekonstrukce pozadí ve videu. Větší prostor je věnovaný stěžejnímu přístupu modelování pozadí pomocí směsi Gaussových funkcí [12]. Jsou zde rovněž uvedeny metody související se zpracováním obrazu, které bylo nutno nastudovat a implementovat hlavně kvůli pozdějšímu vylepšování Gaussova modelu. V třetí kapitole následuje

zejména implementace postupů v praxi a u každé metody probrané v teoretické části zhodnocení při aplikaci na fotografie. Čtvrtá kapitola ukazuje především výstupy algoritmů. Poslední kapitola (Závěr) zhodnocuje dosažené výsledky a poukazuje na možnost dalšího rozšíření práce.

Program byl napsán v jazyce C/C++ s využitím knihovny pro počítačové vidění OpenCV [1]. Knihovna byla využita zejména pro základní práci s obrazem (načtení obrazu, převod do grayscale apod.), a to hlavně kvůli úspoře mého času i výpočetního výkonu vzhledem k optimalizaci takových funkcí. V další fázi vývoje jsem si ze stejných důvodů dovolil převzít a upravit část otevřeného kódu týkajícího se směsi Gaussových funkcí. Jedná se o konzolovou aplikaci. Dílo je vysázeno typografickým systémem LaTeX.

Diplomová práce navazuje na semestrální projekt, v jehož rámci jsem nastudoval metody na získávání pozadí ve videu a implementoval základní metody jako medián, průměr a modus snímků. Konkrétně se jedná o podkapitoly z teoretické části 2.1, 2.2, 2.8 a 2.9. Dále návrh a diskuzi výsledků základních metod (většina třetí kapitoly kromě části 3.4). Taktéž čtvrtá kapitola obsahuje část výsledků dosažených při práci na semestrálním projektu. Navazující práce se zaměřila hlavně na detailnější prostudování, implementaci a vylepšení pokročilejší metody Gaussových funkcí, která je více invariantní vůči změnám osvětlení, tak jak je požadováno v zadání.

Kapitola 2

Teorie

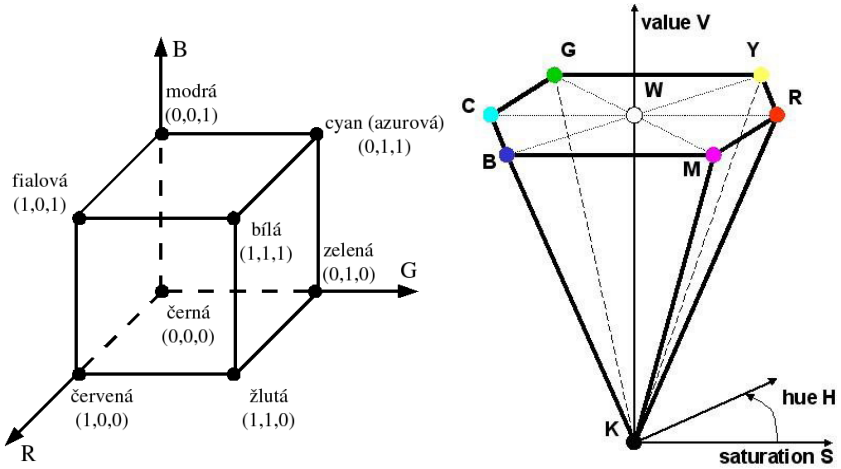
2.1 Barevné prostory

Barevné modely popisují barvy, umožňují s nimi pracovat, míchat je a podobně. Známe jich více druhů, přičemž každý model je přizpůsoben k jinému účelu použití. Práce s barvami pixelů je v tomto textu stěžejní, proto se seznámíme ze základními barevnými modely, u kterých budeme později, v další části práce, zkoumat jejich vhodnost vzhledem ke zvolené implementaci.

2.1.1 RGB

Barevný model RGB [16] neboli červená–zelená–modrá funguje na principu aditivního míchání barev. Je to základní barevný model v oblasti počítačové techniky. Používá se například u monitorů (jedná se o míchání vyzařovaného světla).

Každá barva je udána mohutností tří základních barev o vlnových délkách 630, 530 a 450 nm. Mohutnost se udává buď v procentech, nebo podle použité barevné hloubky jako určitý počet bitů vyhrazených pro barevnou komponentu (pro 8 bitů na komponentu je rozsah hodnot 0–255, pro 16 bitů na komponentu je rozsah hodnot 0–65 535), přičemž čím větší je mohutnost, tím vyšší intenzita zobrazované barevné komponenty.



Obrázek 2.1: Barevný model RGB a HSV

Model RGB je možné zobrazit jako krychli, ve které každá z kolmých os udává škálu mohutností barevných složek. Potom libovolný bod se souřadnicemi (r, g, b) v této krychli udává hodnotu výsledné barvy. Čím větší je součet mohutností, tím světlejší je výsledná barva.

2.1.2 HSV

Barevný model HSV [15] (Hue, Saturation, Value) nejvíce odpovídá lidskému vnímání barev. Model se skládá ze tří složek (nejsou to základní barvy), u nichž je nutno hlídat možné nesmyslné kombinace hodnot. Podobně jako HSL (Hue, Saturation, Lightness) je popsán v cylindrických souřadnicích. Narozdíl od něj má HSV plně syté barvy v hodnotě $v = 1$, kdežto HSL v hodnotě $l = 0{,}5$. Základní složky jsou:

- Hue – převládající barevný tón. Někdy je označován jako odstín barvy. Měří se jako poloha na standardním barevném kole ($0\,^\circ$ až $360\,^\circ$). Obecně se odstín označuje názvem barvy.
- Saturation – sytost barvy. Vyjadřuje příměs jiné barvy. Někdy též chroma, síla nebo čistota barvy. Představuje množství šedi v poměru k odstínu. Měří se v procentech od $0\,\%$ (šedá) do $100\,\%$ (plně sytá barva). Na barevném kole vzrůstá sytost od středu k okrajům. Například červená s 50% sytostí bude růžová.
- Value – hodnota jasu, množství bílého světla. Relativní světlost nebo tmavost barvy. Jas vyjadřuje kolik světla barva odráží. Změnu její hodnoty si můžeme představit jako přidávání černé do základní barvy. Přechod pro hodnotu v od bílé k černé je od 0 do 1, stejně jako pro hodnotu l u HSL.

Převod RGB do HSV:

Hodnoty složek *rgb* jsou reálná čísla mezi 0 a 1. Maximální hodnota se rovná největší z hodnot *r, g* a *b*. Minimální hodnota se rovná nejmenší z těchto hodnot. Následně se spočítají hodnoty *h, s, v* v HSV prostoru, kde $h \in \langle 0, 360)$ je úhel odstínu v mírách, $s \in \langle 0, 1\rangle$ a $v \in \langle 0, 1\rangle$ jsou saturace a světlost. Vypočítáme:

$$
\begin{aligned}
h &= \begin{cases}
\text{nedefinováno}, & \text{if } max = min \\
60\,^\circ \frac{g-b}{max-min} + 0\,^\circ, & \text{if } max = r \text{ a } g \geq b \\
60\,^\circ \frac{g-b}{max-min} + 360\,^\circ, & \text{if } max = r \text{ a } g < b \\
60\,^\circ \frac{b-r}{max-min} + 120\,^\circ, & \text{if } max = b \\
60\,^\circ \frac{r-g}{max-min} + 240\,^\circ, & \text{if } max = g
\end{cases} \\
s &= \begin{cases}
0, & \text{if } max = 0 \\
\frac{max-min}{max} = 1 - \frac{min}{max}, & \text{jinak}
\end{cases} \\
v &= max
\end{aligned}
$$

Převod z HSV do RGB: Se zadanými hodnotami odstínu, jasu a sytosti $(h,\ s,\ v)$ vypočítáme *rgb* hodnoty takto:

$$\begin{aligned} h_i &= \left\lfloor \frac{h}{60} \right\rfloor \mod 6 \\ f &= \frac{h}{60} - h_i \\ p &= v(1-s) \\ q &= v(1-fs) \\ t &= v(1-(1-f)s) \end{aligned} \qquad (r,g,b) = \begin{cases} (v,t,p), & \text{if } h_i = 0 \\ (q,v,p), & \text{if } h_i = 1 \\ (p,v,t), & \text{if } h_i = 2 \\ (p,q,v), & \text{if } h_i = 3 \\ (t,p,v), & \text{if } h_i = 4 \\ (v,p,q), & \text{if } h_i = 5 \end{cases}$$

2.1.3 Grayscale

Grayscale je obraz ve stupních šedi (viz obr. 3.1). Nejčastěji se používá redukce z barevného 24 bitového RGB modelu. Stupně šedi jsou většinou vzorkovány na 8 bitů, což dává standardních 256 úrovní šedi. Pro převod s RGB do grayscale se používá empirický vztah.

$$I = 0.299R + 0.587G + 0.114B \qquad (2.1)$$

2.2 Vzdálenostní funkce

Často potřebujeme od sebe rozlišit dva objekty. Jejich podobnost, případně odlišnost můžeme vyjádřit pomocí vzdálenosti [17]. U práce s obrazem je nutnost rozlišit podobnost dvou pixelů. V závislosti na barevném modelu roste dimenze objektů, které mezi sebou porovnáváme (šedotónový obraz = 1 dimenze, RGB obraz = 3 dimenze, RGB obraz + pozice xy = 5 dimenzí).

Euklidovská vzdálenost mezi dvěma body je běžná vzdálenost, kterou známe z Pythagorovy věty.

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \cdots + |x_{ip} - x_{jp}|^2} \qquad (2.2)$$

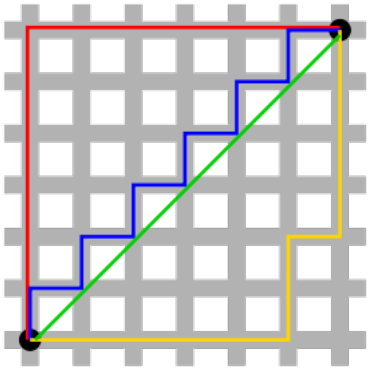
Manhattonovská vzdálenost je pouze suma absolutních hodnot rozdílů souřadnic bodů v prostoru. Pro lepší pochopení toto demonstruje obrázek 2.2.

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{ip} - x_{jp}| \qquad (2.3)$$

Minkowského vzdálenost je zobecnění Euklidovské a Manhattonovské metriky.

$$d(i,j) = (|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{ip} - x_{jp}|^q)^{\frac{1}{q}} \qquad (2.4)$$

kde $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ a $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ jsou dva $p$dimenzionální objekty.



Obrázek 2.2: Manhattonovská a Euklidovská vzdálenost. Červená, modrá a žlutá čára mají stejnou délku (12) a v Manhattonovské vzdálenosti je to vzdálenost dvou černých bodů. Naproti tomu zelená čára znázorňuje Euklidovskou vzdálenost ($6\sqrt{2}$).

2.3 Morfologické operace

Matematická morfologie [22] je teorie a technika k analyzování a zpracovávání geometrických vlastností objektů, např. velikosti, tvaru a vnitřní struktury objektů. Morfologie má uplatnění většinou u digitálního předzpracování obrazu, kde se jedná o různé transformace obrazu podle příslušných operátorů, ale taktéž může být aplikována na grafy, povrchy a další prostorové objekty. Matematická morfologie sloužila původně k zpracování binárního obrazu, ale později byla rozšířena i na šedotónový obraz. Mezi základní operace patří dilatace a eroze.

Morfologická transformace [27] je dána relací mezi obrazem (bodovou množinou X) a typicky menší bodovou množinou nazývanou strukturním elementem B, který je vztažen k lokálními počátku. Aplikace morfologické transformace na obraz odpovídá systematickému posunu strukturního elementu po obraze. Výsledek transformace v každé poloze odpovídá relaci. Binární matematická morfologie pracuje s binárním obrazem s definičním oborem $\mathbb{Z}^2$ a oborem hodnot $\{1, 0\}$.

Dilatace [3] pozvolna zvětšuje hranice objektu na popředí a zmenšuje díry v objektu. Používá se pro zaplnění malých děr a úzkých zálivů v objektech. Dilatace má dva vstupy: vstupní obraz a většinou malou matici bodů známou jako strukturní element nebo jádro. Je to právě jádro, které ovlivňuje výsledný efekt dilatace na vstupní obraz.

Dilataci vypočítáme porovnáváním všech pixelů v obraze vůči jádru, které přikládáme středem na vstupní pixel. Pokud některý pixel v jádře odpovídá pixelu popředí, pak je vstupní pixel považován taktéž za popředí. Například pixely pozadí s jádrem 3 x 3 se stanou popředím, pokud sousedí (8–okolí) s některým s pixelů popředí. Takové pixely právě leží na hranách objektu popředí. Má to za následek, že se objekt zvětšuje ve všech směrech (díry se zmenšují).

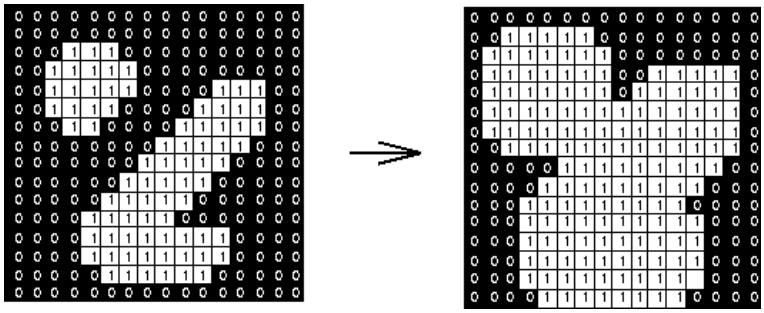
Matematicky zapsáno jako:

$$X \oplus B = \{p \in \mathbb{E}^2 : p = x + b, x \in X \text{ a } b \in B\} \tag{2.5}$$

Lze rovněž vyjádřit jako sjednocení posunutých obrazů:

$$X \oplus B = \bigcup_{b \in B} X_b \tag{2.6}$$

Nejčastěji používaným jádrem je čtvercová matice 3 x 3, ale mohou být využívány i jiné. Větší jádro vytvoří větší dilatační efekt, ale velmi podobné funkčnosti lze dosáhnout i opakováním více dilatací s menším jádrem s podobnými vlastnostmi.



Obrázek 2.3: Dilatace s použitím jádra 3 x 3 pixely

Eroze je duální morfologická operace k dilataci. Odstraňuje hranice objektu na popředí a zmenšuje tím díry v objektu. Používá se ke zjednodušení struktury objektu a rozložení objektu na části. Objekty menší než strukturní element vymizí (například čáry tloušťky 1).

Erozi vypočítáme porovnáváním všech pixelů v obraze vůči jádru, které přikládáme středem na vstupní pixel. Pokud všechny pixely v jádře odpovídají pixelu popředí, pak je vstupní pixel ponechán jako popředí. Uvažujeme-li jádro 3 x 3, efekt operace spočívá v odstranění všech pixelů popředí, které nejsou kompletně obklopeny (8–okolí) dalšími pixely popředí. Takové pixely právě leží na hranách objektu popředí. Má to za následek, že se objekt zmenší ve všech směrech (díry se zvětší).

To v předchozí zavedené notaci znamená:

$$X \ominus B = \{p \in \mathbb{E}^2 : p + b =\in X \text{ pro každé } b \in B\} \tag{2.7}$$

Tj. pro každý bod obrazu p se ověřuje zda pro všechna možná $p+b$ leží výsledek v X. Pokud ano, je výsledek popředí (1), jinak pozadí (0).

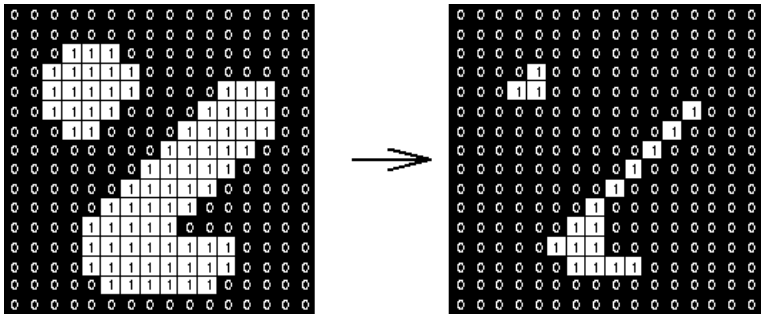
Lze rovněž vyjádřit jako průnik všech posuvů obrazu X o vektor -b:

$$X \ominus B = \bigcap_{b \in B} X_{-b} \tag{2.8}$$

Otevření je definováno jako eroze následovaná dilatací za použití stejného strukturního elementu pro obě operace. Otevření je duální operace k uzavření.

$$\text{opening} = (X \ominus B) \oplus B \tag{2.9}$$

Pokud se obraz nezmění po otevření strukturním elementem B, říkáme, že je otevřený vzhledem k B.



Obrázek 2.4: Eroze s použitím jádra 3 x 3 pixely

Uzavření je definováno jako dilatace následovaná erozí za použití stejného strukturního elementu.

$$\text{closing} = (X \oplus B) \ominus B \tag{2.10}$$

Jedna z vlastností dilatace je vyplnění děr v objektu popředí, a proto tak může sloužit například k potlačení šumu typu pepř. Jeden z problémů je, že dilatace zdeformuje všechny oblasti pixelů bez rozdílů. Uzavření vyplní díry v objektu, a přitom přibližně zachová jeho původní tvar po obvodu. Po jednom uzavření je množina již uzavřená a dalším použitím těchto transformací se již nic nemění.

2.4 Connected component

Jak napovídá název, Connected component labeling je metoda určená k označení částí v obrazu, které k sobě patří. Název je převzat z angličtiny a nemá ustálený český překlad. Metoda je jakási analogie k semínkovému vyplňování. Analýza probíhá procházením celého obrazu pixel po pixelu za účelem identifikace spojených (sousedních) regionů. Tj. regionů složených ze vzájemně sousedících pixelů, které mají stejnou intenzitu. Algoritmus je v počítačovém vidění využíván zejména k detekci osamocených (nespojených) oblastí v binárním obraze, ovšem je možné zpracovávat i šedotónový obraz. Určení sousednosti může probíhat buď nad 4–okolí, nebo nad 8–okolí. V níže uvedeném příkladu se předpokládá binární obraz.

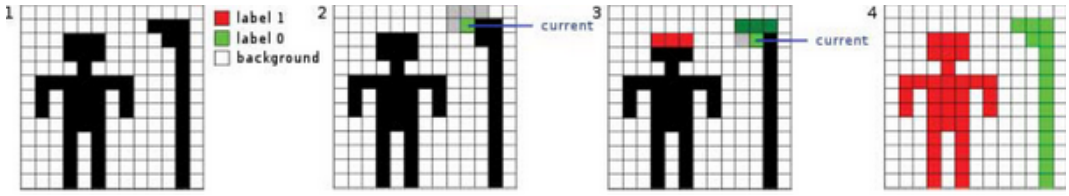
Algoritmus [21] je dvouprůchodový: první průchod slouží k určení ekvivalence a přiřazení dočasného označení a v druhém průchodu se nahradí tyto dočasné značky konečnými značkami, které rozdělí obraz na finální regiony. Vstupní data mohou být zpracovávána *in situ*. První průchod:

- Iteruj přes všechny řádky a sloupce v obrazu
- Pokud není hodnota pixelu pozadí
  - Zjisti sousední pixely v obraze
  - Pokud se žádní sousedé nenalezli, označ zpracovávaný pixel jako unikátní
  - Jinak najdi souseda s nejnižší značkou a přiřaď ji zpracovávanému pixelu
  - Ulož ekvivalenci mezi sousedními značkami

Druhý průchod:

- Iteruj přes všechny řádky a sloupce v obrazu

- Pokud není hodnota pixelu pozadí
  - Přeznač zpracovávaný pixel nejnižší ekvivalentní značkou



Obrázek 2.5: Postup dvouprůchodového algoritmu connected component na binární obraz

2.5 Hledání podobných regionů obrazu

Rekonstrukce pozadí v sobě může obnášet i nutnost najít podobné části v obrazu. Z těchto částí lze posléze vybrat takovou, která by nejlépe odpovídala modelu pozadí. V další části práce jsou představeny metody založeny vždy na statistice výběru podobných částí, ale jedná se o práci nad jednotlivými pixely (tzv. per-pixel operace). Někdy ovšem potřebujeme počítat nad celými regiony a tyto regiony mezi sebou porovnávat. Nejjednodušším způsobem by mohlo být nad těmito regiony vypočítat histogramy a části obrazu porovnávat podle nich. Uvážíme-li, že obraz není nic jiného než 2D signál, nabízí se porovnání těchto signálů korelací.

Korelace slouží [28] k vzájemnému porovnávání dvou navzorkovaných signálů. Jako výsledek získáme posloupnost čísel udávající podobnost signálů, jejich posunutí, periodu apod. Zvláštní případ je autokorelace, kdy se porovnávají dva shodné signály. Ta je z našeho pohledu zajímavější. Algoritmus je založen na vzájemném násobení vzorků signálu a počítání sumy z těchto vzorku.

$$R(m) = \sum_{n=0}^{N-m-1} s(n)s(n+m) \tag{2.11}$$

Normalized cross-corelation má v určitých případech lepší uplatnění než klasická korelace. Nevýhoda standardní autokorelační funkce je postupné zkracování oblasti, ze které autokorelační koeficienty počítáme. Je proto výhodnější přejít na *cross-korelační* funkci. Začátek srovnávání označme zr.

$$CCF(m) = \sum_{n=zr}^{zr+N-1} s(n)s(n+m) \tag{2.12}$$

Při výpočtu CCF můžeme narazit na problém příliš velké energie jednoho ze signálů, která přebije podobnost dvou rámců. Rozdílnost energií originálního a posunutého rámce řešíme pomocí normalizace.

Tím dostáváme *normalizovanou cross-korelaci*:

$$NCCF(m) = \frac{\sum_{n=zr}^{zr+N-1} s(n)s(n+m)}{\sqrt{E_1 E_2}} \tag{2.13}$$

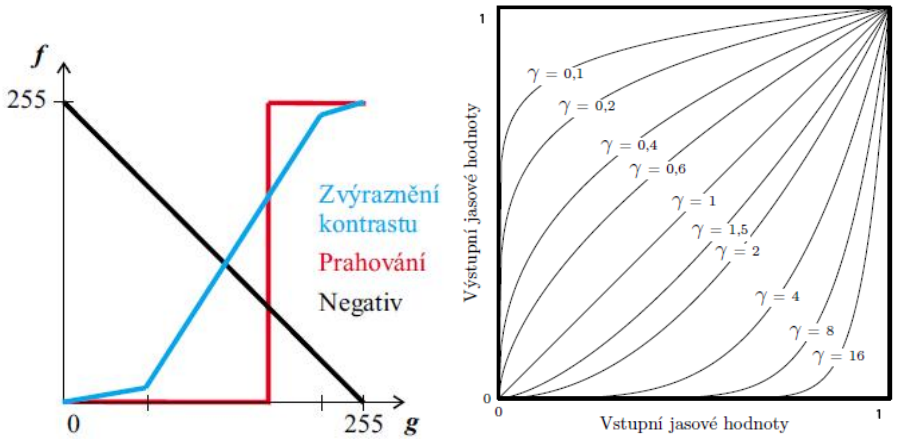
kde E_1 a E_2 jsou energie originálního a posunutého rámce:

$$E_1 = \sum_{n=zr}^{zr+N-1} s^2(n) E_2 = \sum_{n=zr}^{zr+N-1} s^2(n-m) \tag{2.14}$$

Template matching je technika hledání vzoru v obraze. Základní myšlenka spočívá v porovnávání všech pixelů v obrázku se vzorem. Vzor je malý výřez z obrázku, který po něm posouváme pixel po pixelu. Pro středový pixel výřezu dostáváme koeficient podobnosti. Způsob jakým se mezi sebou výřez a obrázek porovnává může být libovolný.

2.6 Jasové transformace

Pořízená sekvence fotografií se může vzájemně jasově lišit, a to především z důvodu změny globálního osvětlení ve scéně (například zatažení oblohy). S touto skutečností se umíme vyrovnat pomocí jasových transformací. Tyto bodové transformace se někdy nazývají *LUT (look-up-table)* transformace, protože v případě diskrétního obrazu může být mapování provedeno pomocí převodní tabulky.



Obrázek 2.6: Vlevo: transformace vstupní jasové stupnice $g(i)$ na výstupní stupnici $f(i)$ [7]. Vpravo: průběh transformační jasové funkce gamma korekce.

Lineární snížení a zvýšení jasu lze zařídit transformační rovnicí $I'(x,y) = kI(x,y)+q$. Máme zde, ale dvě neznámé, a tak můžeme jas buď upravit přičtení/odečtení konstanty q, nebo vynásobením konstantou k. Lepší řešení je násobení konstantou, protože se v obraze zčásti zachová i kontrast. Pouhé přičtení/odečtení konstanty pouze posune histogram jedním či druhým směrem a výsledek pak působí nepřirozeně. Samotné násobení konstantou $I'(x,y) = kI(x,y)$ je možno rozdělit na dva případy

(snižování a zvyšování jasu), jak je to řešeno v [5], aby se předešlo ořezávání hraničních hodnot černé a bílé. Hodnotou L se většinou označuje počet úrovní jasu. Pro osmibitový obraz platí $L = 256$. Ve vzorci znamená L_{max} maximální hodnotu úrovně jasu (nejčastěji $L_{max} = 255$ nebo $L_{max} = 1$ v případě normalizovaného jasu do rozmezí $< 0, 1 >$). Proměnná p je kladné číslo od 0 do 100 a udává o kolik procent se má jas zvýšit/snížit. Průběh funkcí je vidět na obrázcích č. 2.7.

Lineární zvýšení jasu:

$$I'(x, y) = L_{max} - (1 - \frac{p}{100})(L_{max} - I(x, y)) \tag{2.15}$$

Lineární snížení jasu:

$$I'(x, y) = (1 - \frac{p}{100})I(x, y) \tag{2.16}$$

Gamma korekce nebo zkráceně gamma může být taktéž použita pro nelineární změnu jasu [23]. S její pomocí se upraví střední části rozsahu intenzity a nedojde tak k přepalům bílé a ztrátě informace u černé barvy.

Nelineární změnu jasu pomocí gamma korekce provedeme:

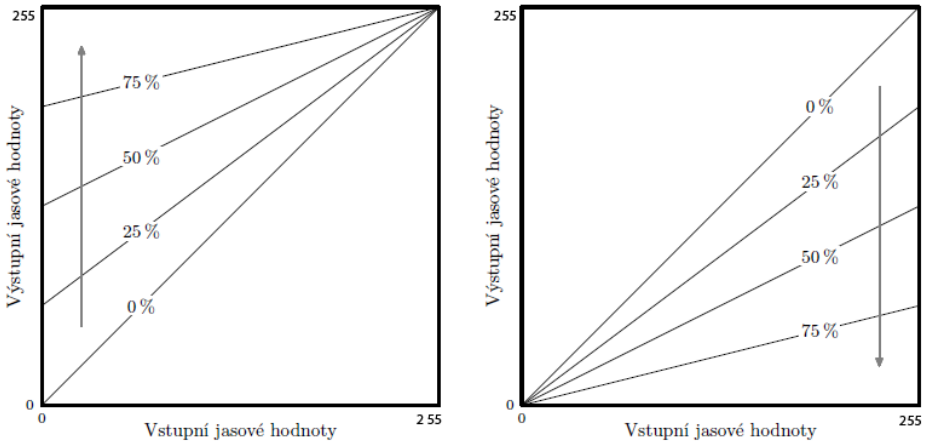
$$I'(x, y) = I(x, y)^{\gamma} \tag{2.17}$$

Průběh funkce ovlivňuje hodnota γ, tak je to ukázáno na obrázku 2.7. Snížení jasu odpovídá hodnota $\gamma > 1$ zvýšení jasu nastane pro hodnoty $\gamma < 1$. Intenzita obrazu musí být normalizovaná do rozmezí $< 0, 1 >$.

Z historických důvodů panuje kolem termínu gamma korekce trošku zmatek. Původně termín Gamma označoval nelinearitu zobrazovacích zařízení (monitorů). Všechny tehdejší zařízení (nejčastěji CRT monitory) fungovaly nelineárně. To znamená, že když se zvedlo napětí na dvojnásobek, nezapříčinilo to zvednutí intenzity vyřazovaného jasu na dvojnásobek. Obraz by měl vždy obsahovat lineární data (např. výstup z fotoaparátu), a proto, aby se na monitoru zobrazily správně, bylo nutné použít mocninu gamma. Každý monitor může mít odezvu jinou, a tudíž i jiný parametr gamma. Většinou závisí jas na napětí exponenciálně, obvykle $\gamma = 2{,}2$. Novější pozměněná definice gamma korekce se ve zpracování obrazu chápe jako nelineární změna jasu aplikovaná na data obrazu. Při transformaci nedochází k ořezání krajních hodnot a navíc má lidské vnímání taktéž nelineární charakter, a proto se tato funkce často využívá. Koneckonců křivka nemusí mít jen jeden parametr gamma, ale může být i různě modifikovaná do libovolného tvaru.

Logaritmický operátor je další nelineární transformační funkcí na změnu jasu a dynamického rozsahu šedotónového obrazu [3]. Dynamický rozsah obrázku je zmenšen vyměněním každé hodnoty pixelu za hodnotu svého logaritmu. To má za následek, že se více zvýší malé hodnoty intenzity jasu. Průběh funkce je podobný gamma korekci pro $\gamma < 1$.

Je experimentálně dokázáno, že vnímání jasu lidským okem má logaritmický průběh [13]. Tato skutečnost byla vědci změřena u jednoduchých zvířecích očí - jejich neurony citlivé na světlo reagují na změnu s logaritmem energie příchozího světla. U lidí se takovéto experimenty neprováděly, ale tvrzení se dokázala za pomocí štítků s rozdílným jasem, které se předkládaly pozorovateli a ten měl říci, jestli se vůči sobě viditelně liší



Obrázek 2.7: Transformační jasové funkce. Po řadě lineární zvýšení jasu a lineární snížení jasu. Obrázky převzaty z [5].

referenční štítek s předloženým štítkem. Pokusy se po krocích provedly pro všechny intenzity od nejsvětlejší po nejtmavší. Hodnoty kroků byly vyneseny do grafu jako funkce intenzity referenčního štítku. Výsledná křivka byla velmi podobná logaritmu.

Většina implementací používá buď přirozený logaritmus, nebo logaritmus o základu 10. Přesto základ logaritmu neovlivňuje tvar křivky, ale pouze změnu měřítka výstupních hodnot. Inverzním operátorem je exponenciální operátor.

Logaritmická mapovací funkce je dána:

$$I'(x,y) = c\log(|I(x,y)|) \tag{2.18}$$

Protože není logaritmus definován pro 0, spousta implementací přidává před logaritmováním hodnotu 1.

$$I'(x,y) = c\log(1+|I(x,y)|) \tag{2.19}$$

Konstanta c měnící měřítko výstupu je pro osmibitový obraz vybrána tak, aby maximální výstupní hodnota byla 255. Pokud označíme maximální hodnotu ve vstupních datech jako R, pak:

$$c = \frac{255}{\log(1+|R|)} \tag{2.20}$$

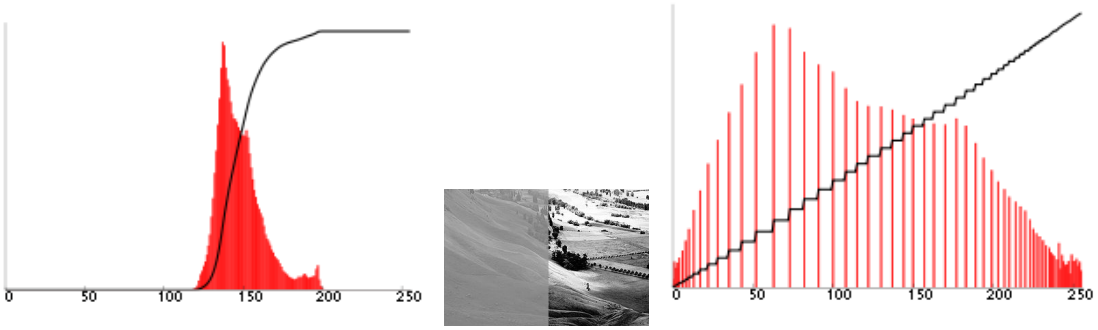
Stupeň logaritmické transformace (zahnutosti křivky) je kontrolován rozsahem vstupních hodnot.

Ekvalizace histogramu [24, 3] je nelineární mapovací metodou na modifikaci dynamického rozsahu a kontrastu u obrazu tak, aby bylo rozložení intenzity v obraze rovnoměrné a přibližně se stejnou četností. Metoda se tedy snaží o úplné využití jasové stupnice. Tato technika může být využita při automatickém porovnávání snímků, protože obraz jasově normalizuje.

Ekvalizaci provedeme:

$$I'(i) = \frac{K(i)}{M \cdot N} * L \tag{2.21}$$

Kde L_{max} je maximální hodnota úrovně jasu (nejčastěji $L_{max} = 255$ nebo $L_{max} = 1$ v případě normalizovaného jasu do rozmezí $< 0, 1 >$). Součin proměnných $M \cdot N$ představuje rozměry obrazu a $K(i) = \sum_{j=0}^{i} H(j)$ je kumulativní histogram a $H(j)$ je $j$tá složka vektoru histogramu H.



Obrázek 2.8: Obraz před a po ekvalizaci histogramu – uprostřed. Vlevo je znázorněn histogram před ekvalizací (červeně) a kumulativní histogram (černě) původního obrazu. Vpravo jsou vidět ekvalizované histogramy.

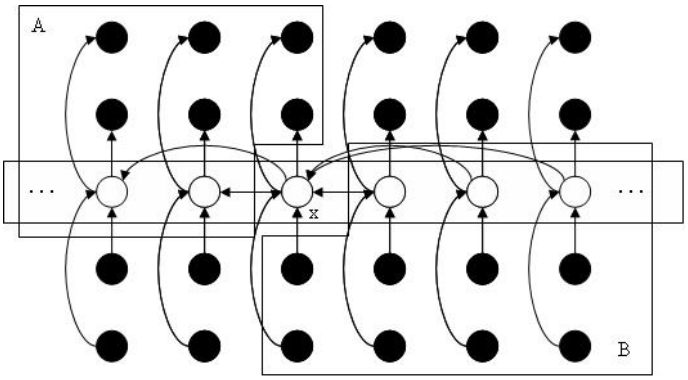
2.7 Medián v lineárním čase

Pro medián platí, že nejméně 50 % hodnot je menších nebo rovných a nejméně 50 % hodnot je větších nebo rovných mediánu. Jinými slovy to znamená, že hledáme $k$tý nejmenší prvek v souboru n čísel, kde k je rovno $\frac{n}{2}$. Standardním přístupem hledáním mediánu bylo až do roku 1972 nejprve soubor seřadit a vybrat prostřední prvek [25]. Seřazení posloupnosti lze realizovat nejlépe se složitostí $O(n \log n)$. Ve skutečnosti můžeme medián vybrat v lineárním čase pomocí algoritmu s výběrem. Tímto algoritmem jsme schopni vybrat $k$tý nejmenší prvek, tj. např. minimum, maximum nebo medián. Pokud tedy nepotřebujeme pole nutně seřadit a stačí nám pouze zjistit medián, lze použít právě selekci, a to i v nejhorším možném případě s lineární složitostí.

Princip selekce je založený na podproceduře quicksortu nazývané rozdělování *partition*. Algoritmus rozdělí a uspořádá soubor hodnot na dvě části kolem zvolené hodnoty – pivota. Nalevo budou přesunuty hodnoty menší nebo rovno než pivot a napravo větší nebo rovno než pivot. Quicksort rekurzivně řadí obě větve seznamu, a to vede na složitost $O(n \log n)$. U selekce nemusíme zkoumat obě větve, protože po rozdělování leží pivot na svém $k$tém místě, a tak stačí rekurzivně provést další rozdělení pouze na větvi, na které předpokládáme medián $k = \frac{n}{2}$, dokud tento $k$tý prvek nebude nalezen. Například pokud má seznam 99 hodnot, tak jako medián hledáme 50tý nejmenší prvek. Když po rozdělování vyšlo, že čísel menších jak pivot L je 70, pak spustíme rozdělování na větvi L, kde hledáme 50tý nejmenší prvek v L. Naopak pokud výjde, že větev L má pouze 40 hodnot, pak spustíme rozdělování na větvi větších hodnot R, kde hledáme $50 - 40 - 1 = 9$tý nejmenší prvek. Tento algoritmus je často nazýván quickselect nebo randomized-select. Funkce vede na průměrnou složitost $O(n)$ a podobně jako quicksort má velmi dobrý výkon v praxi. Naneštěstí je stejně jako quicksort náchylná na správnou volbu pivota a při neustále špatném výběru pivota může algoritmus degradovat v nejhorším možném případě až na složitost $O(n^2)$. Řešením je zajistit výběr vhodných pivotů, který by zaručil správným rozdělováním lineární složitost.

Takovým pivotem je medián mediánů.

Algoritmus nejprve rozdělí seznam n prvků do skupin po pěti, a z každé skupiny selekcí vybere medián a uloží do nové skupiny. Z nově vzniklé skupiny $\frac{n}{5}$ čísel opět selekcí vybere medián. Tento medián mediánu je zvolen za pivota. Zvolený pivot je menší/větší nebo rovno polovině prvků ze seznamu mediánů, což je $\frac{n}{10}$ prvků na každou polovinu. Každý z těchto prvků je medián z 5 hodnot, tzn. že 2 hodnoty jsou menší/větší nebo rovno tomuto mediánu. Proto je pivot menší než $\frac{3n}{10}$ a větší než $\frac{3n}{10}$ všech hodnot. Pivot tak rozdělí seznam někde mezi 30–70 %, a to zajistí i v nejhorším případě lineární složitost. Důkaz, že složitost takového řešení je opravdu $O(n)$, lze najít například v [25, 4].



Obrázek 2.9: Schéma vybrání medianu mediánů. Levá skupina $A <= \frac{3n}{10}$, pravá skupina $B >= \frac{3n}{10}$. Počet n prvků je reprezentovaný kruhy. Pole je rozděleno na skupiny po 5 prvcích. Mediány každé skupiny jsou bílé kruhy a median mediánů je označen jako x. Šipky vedou od prvků s větší hodnotou k menším. Odtud jde vidět, že 3 z 5 prvků vpravo od x jsou větší nebo rovno než x – skupina B a 3 z 5 prvků vlevo od x jsou menší nebo rovno než x – skupina A.

2.8 Odečítání pozadí ve video sekvenci

V různých dohledových systémech je detekce pohybu velmi důležitá. Například monitorování dopravy (počítání nebo identifikace vozidel), sledování chování osob a podobně. V takovýchto systémech se většinou jedná o napevno umístěnou kameru s neměnnými parametry.

Jak ukazuje obrázek 2.10 ve videu jsou středem zájmu především objekty na popředí, nicméně abychom je nalezly, musíme nejprve ve videu rozlišit pozadí. Proto lze inspiraci hledat i v algoritmech vyvinutých pro video, přestože jsou výchozí podmínky pro rekonstrukci pozadí z fotografií a videa rozdílné. Přehled různých metod je přehledně popsán v materiálech [14] k přednášce věnované tomuto tématu. Odtud jsem si taktéž vypůjčil některé obrázky.

Nejrozšířenější úloha u videa je detekce pohybu nebo pohybujícího se objektu na popředí. V zásadě se dají úlohy ve videu rozdělit do dvou skupin: metody sledující pohyb ve videu bez znalosti pozadí a metody vytvářející si model pozadí. Mezi zástupce první skupiny patří například metoda porovnávání rozdílů snímků (frame difference), která pouze odečítá snímky od sebe a určuje zda-li v sekvenci nedošlo ke změně, a tím pádem k pohybu. Další možností je mezi sebou porovnávat histogramy obrazu nebo LBP koeficienty.

Obrázek 2.10: Typický problém jak ze vstupních dat (vlevo) extrahovat popředí (vpravo)

Do druhé skupiny patří odečítání pozadí (background substraction). To zahrnuje spočítání referenčního obrazu (způsobů existuje celá řada), od něj odečtení každého nového snímku a prahování výsledku.

U videa se dá rovněž využít optický tok, a to buď sám o sobě, nebo jako vylepšení některého stávajícího algoritmu. Optický tok je používán na detekci a sledování objektů ve videu. Metoda je teoreticky schopna poradit si i se současným pohybem kamery [20]. Využití optického toku může znamenat velký přínos pro vytváření modelu pozadí, jelikož dokáže odhalit nebo dokonce predikovat pohyb a vyřadit hýbající se objekty. Na druhou stranu je optický tok náchylný na šum.

2.8.1 Základní metody

Základní přístup je, že aktuální snímek $I(x, y, t)$ v čase t odečteme od snímku pozadí $B(x, y, t)$ v čase t, a pokud by rozdíl byl větší než zvolený práh, jedná se o popředí.

$$|\text{snímek} - \text{pozadí}| > Th \tag{2.22}$$

Obrázek 2.11: Odečtením aktuálního snímku (vlevo) od vytvořeného pozadí (vpravo) získáme popředí

Zde ovšem narážíme na hlavní problém, a to jak správně určit referenční snímek pozadí? Různé metody se s tím vypořádávají různě. Od náhodného výběru snímku pozadí až po statistické počítání nejpravděpodobnějšího snímku. Mezi základní metody se dá zařadit:

Rozdíl snímků si neláme hlavu se zjišťováním snímku pozadí, prostě nějaký snímek jako pozadí určí (například předchozí snímek). To můžeme popsat rovnicemi:

$$B(x, y, t) = I(x, y, t - 1) \tag{2.23}$$

$$|I(x, y, t) - I(x, y, t - 1)| > Th \tag{2.24}$$

Funkce $B(x, y, t)$ a $I(x, y, t)$ znamenají po řadě bod pozadí a bod obrazu na souřadnicích x, y v čase t. Tento přístup je úplně základní a nedává dobré výsledky. Je velmi citlivý na strukturu objektů, jejich rychlost a snímkovací frekvenci.

Průměr snímků předpokládá, že velkou dobu převažuje ve videu obraz pozadí. Pozadí se tedy určí jako aritmetický průměr předchozích n snímků.

$$|I(x, y, t) - \frac{1}{n} \sum_{i=0}^{n-1} I(x, y, t - i)| > Th \tag{2.25}$$

Další alternativou by byl vážený průměr, kde novější snímky mají větší váhu než starší apod.

Medián předchozích n snímků vychází ze stejného předpokladu. Díky své povaze však většinou dává lepší výsledky.

$$|I(x, y, t) - median(I(x, y, t - i))| > Th \tag{2.26}$$

kde

$$i = \{0, \ldots, n - 1\}$$

Četnost pixelů počítá stejné výskyty intenzit pixelů. Za pixel pozadí určí ten, který se v sekvenci vyskytoval nejčastěji. Metoda tedy spoléhá na to, že právě pozadí bude zastoupeno na snímcích nejvíce. Na podobném principu funguje další algoritmus, který naopak předpokládá, že pozadí se na snímcích vyskytuje nejdéle. Tzn. porovná **časové úseky**, kdy se intenzita pixelů nemění a vybere ten nejdelší.

Výhodou těchto řešení je jednoduchá implementace a rychlost. Mezi nevýhody patří, že přesnost závisí na počtu snímků za sekundu a rychlosti objektů. Další nevýhodou je poměrně větší paměťová náročnost u mediánu a průměru. To se ovšem dá odstranit takzvaným klouzavým průměrováním:

$$B(x, y, t) = \frac{t - 1}{t} B(x, y, t - 1) + \frac{1}{t} I(x, y, t) \tag{2.27}$$

nebo

$$B(x, y, t) = \alpha I(x, y, t - 1) + (1 + \alpha) B(x, y, t - 1) \tag{2.28}$$

kde α je učící koeficient, typicky 0,05 Hlavní nevýhodou však je, že práh Th není funkcí času a dále existuje jen jeden globální práh pro všechny pixely v obrázku.

2.8.2 Pokročilé metody

Pokročilejší metody se lépe vyrovnávají se změnami osvětlení. Většinou nemají napevno nastavenou hodnotu prahování. Ta se může měnit s časem nebo se používá více prahů. Dalším důležitým rozdílem oproti základním metodám je samotný přístup k modelování pozadí. Většinou si umí poradit s multimodálním pozadím. Tím se myslí, že pozadí se skládá z více barev. Například větev pohybující se ve větru bude působit častou změnu intenzity. Když algoritmus dokáže uvážit, že se pozadí může skládat z více složek (větev a pozadí za větví), potom je schopen lépe rozklíčovat popředí. Na druhou stranu pokročilejší přístupy většinou překypují řadou parametrů, které musí být nastaveny s rozvahou.

Metody které lze považovat za pokročilé se mohou skládat z jednodušších metod, a taktéž v sobě zpravidla zahrnují předzpracování snímků a finální doladění výsledku. Pro představu si některé z nich uvedeme.

Shlukovací algoritmy se snaží najít shluky pixelů, které s největší pravděpodobností odpovídají pozadí. Jedním z takových algoritmů pracující v reálném čase je online clustering [26]. Jako shlukovací algoritmus je zde použit K-means. Metoda předpokládá, že pozadí nebude ta část, která se vyskytuje ve videu krátkou dobu. Algoritmus funguje ve třech krocích.

- V prvním kroku se příchozí pixel přiřadí na základě vzdálenosti do nejbližšího clusteru (středu clusteru). Přiřazení je podmíněno splněním kritéria, že vzdálenost je menší než zvolený práh. Po zařazení do shluku je přepočítán střed clusteru a počet pixelů v clusteru. Když se pixel do zvoleného prahu nevejde, nepřipadá tak do žádného clusteru, a proto je vytvořen zcela nový shluk se středem v příchozím pixelu.
- Druhý krok následuje po skončení shlukování a spočívá v selekci shluků. U každého shluku je spočítána pravděpodobností výskytu odvíjející se od počtu položek v shluku a shluky s nejmenší pravděpodobností výskytu jsou odstraněny.
- Posledním krokem je konečné vybrání intenzity pozadí. Po selekci v předchozím kroku se znovu přepočítá pravděpodobnost a za pozadí se určí shluky s největší pravděpodobností. Kolik shluků bude modelovat pozadí je opět dáno parametrem. Výsledkem je multimodální pozadí - je popsáno více shluky.

W^4 [6, 10] je přístup, ve kterém je pixel označen jako popředí jestliže:

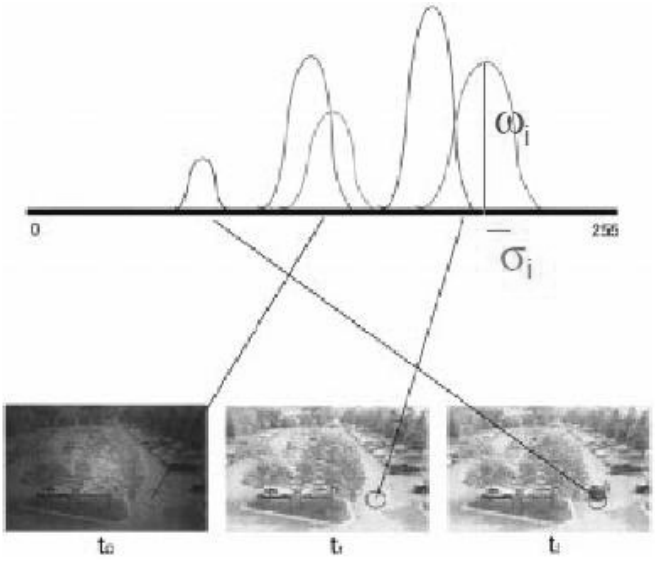
$$|M - I_t| > D \text{ nebo } |N - I_t| > D \tag{2.29}$$

kde per-pixel parametry M, N a D reprezentují minimum, maximum a největší absolutní rozdíl mezi snímky pozorovaný v pozadí. Tyto parametry jsou spočítány při inicializaci algoritmu a posléze samozřejmě periodicky aktualizovány. Konečné popředí je filtrováno erozí kvůli eliminací mezer o velikosti jednoho pixelu a sloučeno pomocí connected component. Nakonec zbylé větší regiony jsou upraveny dilatací a erozí.

Adaptive background mixture model [12] popisuje každý pixel jako směs Gaussových funkcí (odtud název metody). Postupně je ze všech snímků obrazu vytvořen, pro každý pixel, model historie složený z K Gaussových rozdělení pravděpodobnosti (obvykle 3–5). Každé rozložení je ohodnoceno váhou. Idea je, že pozadí by mělo nejvíce odpovídat nejvyšší váze, respektive sumě nejvyšších vah. Když totiž nebude váha jedné Gaussovy funkce dost velká, bude se pozadí modelovat více funkcemi, dokud se nedosáhne

požadované hodnoty váhy. Takto se právě postihne princip opakujícího se pozadí, například již zmíněné kmitající větvičky. Problém multimodálního pozadí byl popsaný v úvodu pokročilejších metod. Pozadí za větví nebude mít samo o sobě dost velkou váhu, protože bude často střídáno onou větví. Obě složky dohromady však již budou mít natolik velkou váhu, že spolehlivě popíšou pozadí. Vše ostatní, co by se dostalo do obrazu, bude popředí.

Ve výčtu jsme si pouze uvedli pouze některé metody, a to spíše pro ilustraci rozdílných přístupů pro popis statistických dat. Poslední zmíněný postup považuji za tolik zajímavý, že jsem se mu rozhodl věnovat celou následující podkapitolu. Přístupů však existuje celá řada. Z dalších například Non-parametric model, Linear predictive filter, Kernel density estimator, Wallflower, atd.



Obrázek 2.12: Ilustrace použití Gaussian mixture model na reálných datech. Příklad převzat z [11].

2.9 Adaptive background mixture model

Místo snahy namodelovat všechny hodnoty pro konkrétní pozici pixelu nějakým jedním typem distribuční funkce, rozhodli se autoři [12] jednodušeji modelovat příslušný pixel jako směs Gaussových funkcí. Na základě stálosti jednotlivé Gaussovovy funkce a její odchylky se určí, jestli pixel může příslušet pozadí. Hodnota pixelu, která nezapadá do rozložení pozadí je považována za popředí, dokud se nevyskytne Gaussian (zahrnující tuto hodnotu) s dostatečně významnou váhou, která by výsledek zvrátila. Pixely nezapadající do modelu pozadí jsou sloučeny pomocí „connected component“ a mohou tak být sledovány jako jeden objekt snímek po snímku.

Systém je schopen vypořádat se změnami osvětlení, je možné naučit se multimodální pozadí a je schopen postihnout i pomalu se pohybující objekty, protože jejich barva má vůči pozadí velkou odchylku.

2.9.1 Modelování pozadí

Každý pixel ve scéně je modelován pomocí K Gaussových funkcí. Pravděpodobnost, že konkrétní pixel má hodnotu x_n v čase N může být napsána jako

$$p(x_n) = \sum_{j=1}^{K} w_j \eta(x_n; \theta_j) \tag{2.30}$$

kde w_k je váha $k$té Gaussovy funkce a $\eta(x; \theta_k)$ je Normální rozložení $k$té funkce představující:

$$\eta(x; \theta_k) = \eta(x; \mu_k; \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \sum_k^{-1} (x-\mu_k)} \tag{2.31}$$

kde μ_k je střední hodnota a $\Sigma_k = \sigma_k^2 I$ je kovarianční matice $k$té funkce.

Každé rozložení je ohodnoceno váhou w_k. Pozadí by mělo nejvíce odpovídat rozložení s největší váhou a zároveň nejmenší standardní odchylkou σ.

Kdyby se nově příchozí pixely příliš neměnily, standardní metoda pro odhad parametrů Gaussova rozložení je *expactation maximalization* (EM). Naneštěstí v čase se mohou hodnoty rychle měnit a navíc se pro každý pixel v obraze počítá více rozložení. Implementace EM algoritmu by byla příliš výpočetně náročná. Je proto vhodnější použít k začlenění nových dat aproximační učící se pravidla.

Nově příchozí pixel je porovnán s každým rozdělením Gaussových funkcí, a pokud je jeho vzdálenost menší než zvolený práh, je pixel zařazen do jedné z funkcí. Jako optimální práh je navrhována hodnota 2,5 σ_k. Hodnota pixelu po zařazení do funkce samozřejmě celou funkci vychýlí, a proto je nutno přepočítat její parametry σ_k, respektive Σ_k, μ_i a w_k. To provedeme těmito online aktualizačními rovnicemi:

$$\begin{aligned}
w_k^{n+1} &= (1-\alpha) w_k^n + \alpha p(\omega_k | x_{n+1}) && (2.32)\\
\mu_k^{n+1} &= (1-\alpha) \mu_k^n + \rho x_{n+1} && (2.33)\\
\Sigma_k^{n+1} &= (1-\alpha) \Sigma_k^n + \rho (x_{n+1} - \mu_k^{n+1})(x_{n+1} - \mu_k^{n+1})^T && (2.34)\\
\rho &= \alpha \eta(x_{n+1}; \mu_k^n, \Sigma_k^n) && (2.35)\\
p(\omega_k | x_{n+1}) &= \begin{cases} 1, & \text{když } \omega_k \text{ je první shoda s Gaussovou funkcí} \\ 0, & \text{jinak} \end{cases} && (2.36)
\end{aligned}$$

kde ω_k je $k$tá Gaussova distribuce, $1/\alpha$ je časová konstanta označující změnu. Parametr $\alpha = 1/\text{velikostOkna}$. Velikostí okna se má mysli počet snímků, ze kterých se vyhodnocuje model.

Když není možné pixel nikam zařadit, nahradí se nejméně pravděpodobná Gaussova funkce novou funkcí se středem v dosud nezařazeném pixelu. Takto vzniklé funkci se nastaví velký rozptyl σ_i a malá váha μ_i.

Ohodnocení Gaussových funkcí, tzv. *fitness* hodnota, je spočítáno jako podíl $\frac{w_k}{\sigma_k}$. Ten rovněž slouží k seřazení Gaussových funkcí podle toho, jak dobře modelují pozadí. Samotné rozdělení na pozadí a popředí určuje seřazení funkcí. Podle přednastaveného prahu se spočítá kolik prvních B Gaussových rozložení bude modelovat pozadí. Ty zbylé pak popisují popředí. Počet funkcí B se spočítá jako:

$$B = \arg\min_{b}(\sum_{j=1}^{b} w_j > T) \tag{2.37}$$

Práh T je měřítko pro minimální zastoupení dat považovaných za pozadí. Jinými slovy určuje minimální pravděpodobnost, že je pozadí obsaženo ve scéně. Malá hodnota T bude mít za následek unimodální pozadí (bude obsahovat jen jedinou distribuci – barvu). Větší hodnota T umožní multimodální pozadí. Tzn. pozadí je složeno ze dvou nebo více barev a umožní postihnout opakující se prvky ve scéně (listí a větve na stromě a podobně).

Známe-li funkce modelující pozadí, respektive popředí, je již snadné určit kam spadá pixel s konkrétní hodnotou. Finální prohlášení pixelu za popředí znamená, že je tato hodnota vzdálená o 2,5 σ_i od každé z B Gaussových funkcí. Oblasti popředí, které k sobě patří je vhodné sloučit například pomocí „2D connected component analysis“, viz sekce 2.4. Vzniknou tím regiony, u nichž lze poté dále určovat velikost pozici, tvar a podobně. Osamocené pixely jsou odstraněny.

Jednou z hlavních výhod metody modelování ze směsí Gaussových funkcí je, že více funkcí v sobě po celou dobu zachovává všechny potenciální modely pozadí. Jakmile se nějaké Gaussovy funkce stanou součástí pozadí, nezničí to současný model. Současný model barvy pozadí zůstane nadále ve směsi, jen bude ohodnocený jako méně pravděpodobný, tzn. nebude se vyskytovat na prvních pozicích v seřazené posloupnosti modelujících funkcí. Například bude-li se ve scéně vyskytovat stacionární objekt dost dlouho na to, aby se stal součástí pozadí a pak se začne pohybovat, distribuce popisující toto pozadí bude stále existovat se stejnými parametry μ a σ^2, ale s nižší váhou w, která brzy zapříčiní odsunutí distribuce na zadnější pozici – nebude již součástí pozadí. Pokud se pohybující objekt znovu vrátí na svou pozici, celkem rychle se na něj algoritmus adaptuje.

2.9.2 Vylepšený online EM algoritmus

Základní verze algoritmu má však i své nevýhody na které bylo upozorněno v [9]. Hlavní nevýhodou je pomalé zotavení z chyby při nevhodné počáteční inicializaci. Když bude prvotní hodnota pixelu obsahovat popředí, bude mít příslušná Gaussova funkce velkou váhu a bude trvat $log_{1-\alpha}(T)$ snímků, než bude právoplatné pozadí správně považováno za pozadí a $log_{1-\alpha}(0,5)$ snímků, než se tato hodnota stane dominantní složkou. Například uvažujme, že se pozadí vyskytuje 60 % času a α je 0,002 (okno je 500 snímků), tak zabere 255 snímků, než bude tato hodnota považována za část pozadí a celých 346 snímků bude potřeba na uznání příslušné Gaussovy funkce jako dominantní. Toto vše může být ještě horší na rušné scéně s méně častým pozadím. Autoři vylepšení rozdělili průběh EM algoritmu na dvě části. Ze začátku se počítá očekávaná statistika podle prvních rovnic, a jakmile je zpracováno prvních L snímků, budou se aktualizační rovnice počítat podle druhých formulí. Tento postup zaručuje dobrý odhad i zpočátku, než je dostupných všech L snímků. Odlišný postup při nenaplněném okně poskytuje v této fázi lepší odhad a umožňuje rychlejší konvergenci při stabilním modelu pozadí. Rovnice počítající s naplněným oknem upřednostňují novější data, a proto se může systém rychleji adaptovat na změny v prostředí.

Online EM algoritmus s očekávanou statistikou používanou v první části (ještě není

k dispozici všech L snímků):

$$w_k^{n+1} = w_k^{n+1} + \frac{1}{n+1}(p(\omega_{n+1}|x_{n+1} - w_k^n) \tag{2.38}$$

$$\mu_k^{n+1} = \mu_k^{n+1} + \frac{p(\omega|x_{n+1})}{p(\omega_{n+1}|x_i)}(x_{n+1} - \mu_k^n) \tag{2.39}$$

$$\Sigma_k^{n+1} = \Sigma_k^{n+1} + \frac{p(\omega|x_{n+1})}{p(\omega_{n+1}|x_i)}((x_{n+1} - \mu_k^n)(x_{n+1} - \mu_k^n)^T - \Sigma_k^{n+1}) \tag{2.40}$$

Druhá část rovnic (již je k dispozici všech L snímků):

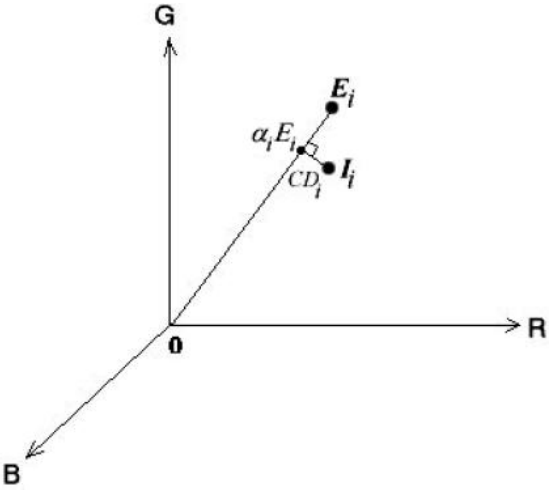
$$w_k^{n+1} = w_k^{n+1} + \frac{1}{L}(p(\omega_{n+1}|x_{n+1} - w_k^n) \tag{2.41}$$

$$\mu_k^{n+1} = \mu_k^{n+1} + \frac{1}{L}\frac{p(\omega|x_{n+1})x_{n+1}}{w_k^{n+1}} - \mu_k^n) \tag{2.42}$$

$$\Sigma_k^{n+1} = \Sigma_k^{n+1} + \frac{1}{L}(\frac{p(\omega|x_{n+1})(x_{n+1} - \mu_k^n)(x_{n+1} - \mu_k^n)^T}{w_k^{n+1}} - \Sigma_k^{n+1}) \tag{2.43}$$

2.9.3 Detekce stínů

Zejména v dohledových systémech vzniká nutnost detekovat stín objektu. Stín se s objektem přirozeně pohybuje, a tak může mást různé detektory pohybu. Jeho barva je navíc natolik odlišná od barvy pozadí, že nebývá správně zařazen a je vyhodnocen jako popředí. Výrazně tak zkresluje siluetu pohybujícího se objektu, a proto je obecně chápán jako problém. Naštěstí však existují způsoby, jak se stínu zbavit.



Obrázek 2.13: Barevný model v RGB prostoru. E_i představuje očekávanou hodnotu $i$tého pixelu a I_i představuje barevnou hodnotu pixelu ve snímku. Rozdíl mezi E_i a I_i je možné rozdělit na jasovou (α_i) a barevnou (CD_i) složku.

Na základě vnímání barevné informace člověkem sestavili pánové Horprasert a spol. [8] barevný model, který matematicky popisuje stín. Důležitým chováním povrchu je, že vlastnosti spektrálního odrazu jsou invariantní vůči změně osvětlení, kompozici scény nebo

geometrii. V Lambertově osvětlovacímu modelu nebo u dokonale matných povrchů je vnímaná barva produktem osvětlení a spektrálního odrazu. Toto je vedlo k myšlence navrhnout barevný model skládající se z barevné a intenzitní složky. Obrázek 2.13 znázorňuje navržený barvený model v třídimenzionálním RGB prostoru. Uvažujme ve fotografii pixel i, potom $E_i = [E_R(i), E_G(i), E_B(i)]$ reprezentuje očekávanou RGB barvu v referenčním snímku nebo snímku pozadí. Úsečka OE_i popisuje očekávaný průběh barevnosti. Dále nechť $I_i = [I_R(i), I_G(i), I_B(i)]$ popisuje RGB barvu pixelu v aktuálním snímku, který chceme odečíst od pozadí. V zásadě chceme změřit posunutí I_i od E_i. Toho dosáhneme rozdělením posunutí na dvě složky, barevné posunutí a intenzitní posunutí.

Intenzitní posunutí (α) Intenzitním posunutím α rozumíme posunutí v jasové složce po úsečce OE_i na barevnostní přímce. Tuto skalární hodnotu vypočítáme minimalizací výrazu:

$$\phi(\alpha_i) = (I_i - \alpha_i E_i)^2 \tag{2.44}$$

α_i představuje velikost jasu pixelu s ohledem k očekávané hodnotě. Když se α_i rovná 1 znamená to, že je jas daného pixelu v aktuálního snímku stejný jako v referenčním snímku. Pokud je α_i menší než 1, je výsledná hodnota blíže k počátku souřadnicového systému a pixel je pak tmavší. Když je α_i větší jak 1, hodnota je světlejší. V analytické geometrii bychom nazvali hodnotu α_i parametrem úsečky OE_i.

Barevné posunutí (CD) Barevné posunutí je definováno jako ortogonální vzdálenost mezi pozorovanou barvou a očekávanou barevnostní úsečkou. Barevné posunutí pixelu i je dáno:

$$CD_i = ||I_i - \alpha_i E_i|| \tag{2.45}$$

Obsáhlejší pojednání včetně rozboru nejrůznějších obrazových charakteristik lze najít přímo v citované práci [8].

Kapitola 3

Automatická rekonstrukce pozadí z fotografií

V této kapitole budou představeny různé metody na získávání pozadí z fotografií. Se základních metod jsem nastudoval a implementoval medián, průměr a modus snímků. Jako pokročilejší algoritmus je představena směs Gaussových funkcí, která si je mimo jiné schopna poradit i se změnami osvětlení.

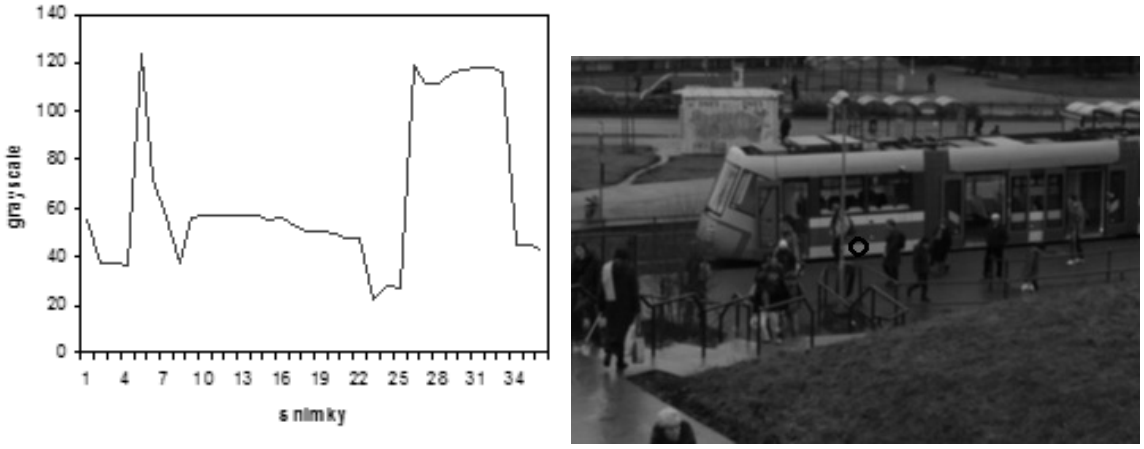
Program je napsán v C/C++ s použitím knihovny OpenCV [1]. Protože knihovna OpenCV podporuje celou řadu funkcí na zpracování obrazu a navíc jsou tyto funkce značně optimalizované, využívám je pro úsporu času i výpočetního výkonu.

Všechny algoritmy byly testovány na shodné sadě dat. Testovací data byly vybrány opravdu citlivě vzhledem k zvolenému problému a měly by tak být schopny prověřit výkonnost algoritmů. Celkem se jedná o 36 fotografií postihující snad všechny problémy, na které je možno narazit při řešení úlohy tohoto typu. Na snímcích jsou zobrazeny jak pohybující se osoby včetně jejich stínů, tak provoz na silnici. Významnou roli hrají tramvaje coby dočasné stacionární objekty, které většinu svého času stojí na zastávce. Všímavý čtenář si jistě povšimne pravých částí obrazů, ve kterých se skutečné pozadí vyskytuje v méně než v padesáti procentech případů. Tento fakt způsobuje testovaným algoritmům velké problémy. V celé scéně se navíc postupně mění globální osvětlení. Pozvolné zatahování oblohy způsobilo, že později pořízené snímky mají vždy celkově menší jas. Pro lepší představu si všimněte obrázku 3.1. V grafu je znázorněna změna intenzity zakroužkovaného pixelu na 28. fotografii. Pro lepší názornost jsou snímky časově seřazeny (není podmínkou), tedy v pořadí, ve kterém byly pořízeny. Intenzita pixelu odpovídající pozadí je na snímcích číslo 1, 7–22 a 34–36. V průběhu fotografování projela zastávkou dvakrát tramvaj, což má v grafu za následek prudkou výchylku intenzity. Tramvaj se objevila poprvé v 5. snímku a podruhé v 26. až 33. snímku, kdy stála na zastávce. Ostatní výchylky intenzity jsou způsobené procházejícími lidmi nebo stínem od další tramvaje v protisměru. Poslední věc, co stojí za zmínku, je (z grafu rovněž patrný) postupný pokles intenzity způsobený pozvolným zatahováním oblohy v průběhu pořizování snímků. Všechny testovací data jsou zobrazeny v příloze textu.

Hlavním omezujícím faktorem při získávání pozadí je určitě množství snímků. Budeme jich mít totiž relativně málo, a přitom požadujeme dobré výsledky. To se nakonec ukázalo jako velký problém. Na druhou stranu nároky na rychlost nejsou zdaleka prioritní, jak tomu může být v extrakci pozadí z videa. Video má sice k dispozici velkou spoustu snímků, nicméně pokud je zde kladen důraz na rychlost algoritmu nebo dokonce na realtime zpra-

cování, jeho výhoda se ztrácí.

U vstupních dat se předpokládá, že budou zobrazovat identickou scénu a že budou pořízeny ze stativu. Při fotografování je vhodné využít dálkovou spoušť. Úplně tak předejdeme rozhýbání fotoaparátu při manuálním stisku spouště. Mělo by se tudíž zamezit tomu, aby si jednotlivé pixely vzájemně neodpovídaly.



Obrázek 3.1: Průběhu změny intenzity na jednom pixelu ve 36 snímcích. Sledovaný pixel je na černobílé fotografii zakroužkován.

3.1 Průměr snímků

Průměr snímků by se dal považovat za úplně základní metodu na získání pozadí. Ve skutečnosti se s ním můžeme setkat i v praxi. Tím ovšem není myšleno použití nějakého softwaru, ale manuální focení s dlouhými expozičními časy. Neměnná scéna zůstane zachována a pohybující objekty budou rozmazány. V případě velmi dlouhé doby expozice objekty úplně zmizí. Fotoaparát vlastně průměruje scénu.

Pozadí získáme tak, že pro každou jednu pozici pixelu v obraze provedeme zprůměrování přes všechny pořízené snímky. V případě, že pracujeme s barevným snímkem se tím rozumí průměr jednotlivých barevných kanálů zvlášť. Jinými slovy intenzita pixelu I na souřadnicích x, y ve výsledném obrazu vznikne jako průměrná hodnota intenzit pixelů ze všech n fotografií. To můžeme popsat rovnicí:

$$I(x,y) = \frac{1}{n}\sum_{i=1}^{n} I(x,y,i) \tag{3.1}$$

Výhodou tohoto řešení je bezesporu jednoduchost a rychlost. Jeho úspěch ovšem hodně záleží na počtu snímků. Čím více snímků, tím bude pozadí kvalitnější a bez duchů. Teoreticky může dávat dobré výsledky, ale pouze za předpokladu, že by bylo k dispozici velké množství snímků a objekty by byly pokud možno co nejvíc pohyblivé. Za další výhodu by se dala považovat hladkost snímku. I chybný výsledný snímek nepůsobí příliš rušivě, protože se u něj neobjevuje výstřelový šum. Ovšem všechny tyto výhody převáží fakt, že průměrování snímků nedává uspokojivé výsledky. Příliš totiž spoléhá na to, že pozadí tvoří výraznou část scény. Výsledek použití průměru snímků ukazuje obrázek 4.1.

3.1.1 Implementace průměru snímků

Intuitivním řešením je načtení všech vstupních fotografií do paměti a pak cyklem přes jednotlivé pixely obraz zprůměrovat. To sebou ovšem u více fotografií nese nepříjemnost, že je nutné mít všechny snímky načtené v paměti zaráz. Zejména u více fotografií s větším rozlišením je proto výhodnější alokovat místo pro akumulátor (pole) stejných rozměrů jako je fotografie. Hodnoty pixelů příchozí fotografie se naakumulují do pomocného pole, pak se fotografie z paměti uvolní a místo ní se načte nová. Po posledním snímku provedeme zprůměrování hodnot v akumulátoru, a tak dostaneme výsledek.

3.2 Medián snímků

Princip mediánu je podobný průměrování. Vychází z předpokladu, že pozadí se vyskytuje na pořízené sekvenci nejčastěji. Platí [18], že nejméně 50 % hodnot je menších nebo rovných a nejméně 50 % hodnot je větších nebo rovných mediánu. Medián jak známo vybírá prostřední hodnotu v seřazené posloupnosti. Pokud má posloupnost sudý počet prvků, obvykle se za medián označuje aritmetický průměr hodnot na místech $\frac{n}{2}$ a $\frac{n+1}{2}$.

Výslednou intenzitu pixelu I na souřadnicích x, y spočítáme jako medián intenzit ze všech n fotografií na pozici x, y. Neboli:

$$I(x, y) = median(I(x, y, i)) \tag{3.2}$$

kde

$$i = \{0, \ldots, n-1\}$$

Výhodou je opět jednoduchost a rychlost. Pokud by se opravdu vyskytovalo pozadí v sekvenci fotografií z více jak 50 %, získáme dobré výsledky. Tato situace však nastane pouze teoreticky. V praxi mohou může být vstup jakýkoli. Přesto se dá říci, že na stejných datech medián funguje viditelně lépe než průměrování. Je to logické, protože v potaz bere ty pravděpodobnější hodnoty pozadí. Výsledný obraz je taktéž hladký. Tak jako tak neustále platí, že tato metoda není schopna zajistit dostatečnou filtraci popředí. Výsledek mediánu ilustruje obrázek 4.2.

3.2.1 Implementace mediánu snímků

Medián je operace paměťově relativně náročná, přesto z povahy úlohy plyne, že vstupních fotografií nebudou stovky, a proto není nutné sáhnout k řešení typu klouzavý medián (v předchozím případě průměr). Nicméně oproti průměrování potřebujeme mít k dispozici všechny hodnoty naráz, nikoli jen jednu naakumulovou hodnotu. Velikost pomocného pole by se rovnala velikosti všech načtených fotografií, takže je snažší na manipulaci načíst všechny fotografie naráz, než je postupně vkládat do pomocného pole. Základním přístupem hledání mediánu je seřadit všechny hodnoty ($O(n \log n)$) a z nich vybrat prostřední. Já jsem se rozhodl implementovat hledání mediánu v lineárním čase postupem zvaným quickselect, viz podkapitola 2.7.

3.3 Modus snímků

Modus [19] náhodné veličiny X (označováno jako $Mod(X)$ nebo $\hat{x}$) je hodnota, která se v daném statistickém souboru vyskytuje nejčastěji (je to hodnota znaku s největší rela-

tivní četností). Představuje jakousi typickou hodnotu sledovaného souboru a jeho určení předpokládá roztřídění souboru podle obměn znaku.

Tato metoda spoléhá na to, že pozadí bude vidět na snímcích nejvíc (s největší četností). Výslednou intenzitu pixelu I na souřadnicích x, y spočítáme jako modus intenzit ze všech n fotografií na pozici x, y. Neboli:

$$I(x, y) = mod(I(x, y, i)) \tag{3.3}$$

kde

$$i = \{0, \ldots, n\}$$

Cesty jak spočítat četnosti intenzit jsem vyzkoušel dvě. Prvním způsobem je počítat modus u jednotlivých kanálů zvlášť. V případě 3 kanálového snímku RGB nebo HSV se spočítá nejvíce zastoupená intenzita u každého kanálu. Výsledný pixel pak bude mít barvu složenou ze tří nejvíce zastoupených složek v kanálu. Vizuální výsledky v RGB prostoru dopadly lépe než v HSV prostoru. Druhým způsobem je počítat četnost u jednotlivých pixelů. Znamená to, že pixel je popsán vektorem tří čísel (například kanály R, G, B) a hledáme nejvíce zastoupené pixely. Pixel s nejvyšší četností je posléze vybrán jako výsledný. Z pokusů vyplynulo, že o málo lepší výsledky produkuje modus vektoru RGB než modus jednotlivých kanálů RGB.

Metoda četností dává lepší výsledky jak medián, má však i své nevýhody. Modus totiž nemusí být rozdělením pravděpodobnosti určen jednoznačně (tzn., že se stejnou nejvyšší frekvencí se může vyskytovat více hodnot). V takových případech počítám průměr intenzit. Dalším nedostatkem je, že vektory se porovnávají vzhledem ke zvolenému prahu, který je sám o sobě magická konstanta. Práh totiž není funkcí obrazu. S tímto problémem se však setkáme u každé metody, která vzájemně porovnává pixely na shodu. Aby byla metoda alespoň z části invariantní vůči změně osvětlení, nemůžeme pixely porovnávat na přesnou shodu. Místo toho bereme pixely jako stejné v určitém rozmezí. Příliš nízký práh nebo porovnání na přímou shodu způsobí silné zašumění výsledku. Naopak příliš vysoká hranice porovnání zahrnuje z větší části i popředí. Jak ovlivňuje nastavení prahu výsledek je ukázáno na obrázku 3.2. Hlavním úskalím, ale zůstává samotný předpoklad, že pro správnou funkci tohoto řešení se musí pixely pozadí vyskytovat v sekvenci snímků nejčastěji.

Obrázek 3.2: Výřezy pozadí metodou modus demonstrující vliv nastavení prahu pro jeden kanál (po řadě 5, 10, 15 pixelů).

3.3.1 Vliv barevných prostorů

Protože je počítání četností značně výpočetně náročná operace, hledal jsem způsob, jak výpočtu co nejvíce ulehčit. Místo pro zrychlení výpočtu je skryto v porovnávání dvou pixelů na shodu. Ve skutečnosti se pixely nemohou porovnávat na přímou shodu. Dva pixely se berou jako stejné v případě, že absolutní hodnota rozdílu jejich intenzit je menší jak zvolený práh.

$$I(x, y, i) = I(x, y, j) \text{ když } |I(x, y, i) - I(x, y, j)| < Th \tag{3.4}$$

Pokud je pixel popsán příznakovým vektorem o více složkách (například R, G, B), porovnáváme vzájemně vektory pomocí vzdálenosti. Vzdálenost spočítáme vzdálenostními funkcemi z kapitoly 2.2. Já jsem si konkrétně vybral Manhattonovskou vzdálenost.

Porovnání se taktéž urychlí snížením dimenze vektorů u porovnávaných pixelů. Místo klasického RGB obrazu, kde jsou pixely popsány třídimenzionálním vektorem (R, G, B) jsem provedl experimenty s dvěmi dimenzemi (H, V) i jednou dimenzí (šedotónový obraz).

Výsledky u barevného modelů **RGB** byly podle předpokladu uspokojivé a lepší než u mediánu. Viz obrázek 4.4. Četnost z pixelu popsaných vektorem dopadla taktéž o něco lépe než četnost počítaná z jednotlivých kanálů zvlášť.

Při práci s **HSV** modelem jsem vyšel z předpokladu, že základní barvu udávají pouze dvě složky, a to barevný tón H a sytost barvy S, kdežto poslední složka V pouze udává světlost barvy. Pixel bylo možno poté popsat pouze vektorem (H, S). Ukázalo se, že tento přístup je poněkud nešťastný a nedává dobré výsledky. Viz obrázek 4.3. Výpočet se sice urychlil, ale absence kanálu V způsobila nesprávné shody. Za shodu byly považovány i jasově velmi vzdálené pixely. To se projevilo silným výstřelovým šumem. Dobré výsledky nedává překvapivě ani samotný model HSV, vezmeme-li v potaz všechny tři kanály. Na závěr se hodí říci, že model HSV má jiný rozsah hodnot než RGB. Proto je nutno před samotnou prací hodnoty normalizovat. Já jsem je konkrétně převedl do stejného rozmezí jako má RGB model (0–255), abych mohl při porovnávání použít stejný práh.

Šedotónový obraz nakonec předčil moje očekávání. Porovnání se urychlí třikrát, a přitom dává dokonce stejně dobré výsledky jako RGB obraz. Výsledek porovnání ve stupních šedi je na obrázku 4.5. V podstatě se samozřejmě pořád pracuje s barevným obrazem, pouze porovnávání pixelů se provede ve stupních šedi. Experimentální výsledky beru jako důležitý poznatek, že ztráta barevné informace se na výsledku porovnání výrazně neprojeví. Toho by se dalo mimo jiné využít například i u segmentace barevného obrazu a místo toho segmentovat obraz šedotónový.

3.3.2 Implementace modu snímků

Podobně jako u mediánu snímku musíme mít v momentě výpočtu modu k dispozici všechny hodnoty. Kvůli vysoké výpočetní náročnosti je vhodné pracovat pouze ze šedotónovým obrazem, tím ovšem vzniká nutnost alokovat pomocné pole o rozměrech výška obrazu krát šířka obrazu, do kterého si ukládáme indexy fotografií s nejvyšší četností pro každý pixel. Pomocné pole je nutnost, pokud nechceme mít v paměti šedotónové snímky spolu s barevnými originály. Po zpracování šedotónových snímků jsou znovu načteny barevné originály, které naplní výsledný obraz podle záznamů četností v pomocném poli.

Výpočetní složitost je bohužel vysoká. Obnáší to zjistit, kolik pixelů z n snímků je stejných ($\pm$ tolerance) jako pixel v $i$tém snímku. To znamená provést porovnání každý

s každým, tj. $n * n$ porovnání, respektive $\frac{n*n}{2}$ v případě, že si pamatuji předchozí porovnání. Výpočet roste kvadratickou složitostí. Na vině je zajištění odolnosti algoritmu vůči změnám osvětlení ve scéně. Hodnoty jasu pixelu nemůžeme porovnávat na přímou shodu, ale vždy v rámci dané tolerance. Jinak by se dala maximální četnost zjistit seřazením pole řadícím algoritmem s linearitmickou složitostí a pak jedním průchodem seřazeného pole najít maximální výskyt stejných hodnot za sebou.

3.4 Mixture of Gaussian

Posledním implementovaným algoritmem je modelování pozadí za pomocí směsi Gaussových funkcí (mixture of Gaussian – MOG). V zadání bylo požadováno nastudovat algoritmus, který si bude schopen poradit i se změnou osvětlení ve scéně. Tuto podmínku splňuje právě MOG. Algoritmus je sice navržen na práci s videem, ale princip lze využít i k rekonstrukci pozadí z fotografií. Jako základ programu slouží postup napsaný podle [9] a představený v teoretické části. Ten má oproti původně navrženému postupu vylepšení v rychlejší konvergenci ke správnému výsledku a rychlejšímu zotavení při chybné počáteční inicializaci, viz kap. 2.9.2. Je tudíž více vhodný k experimentům s méně snímky.

MOG dává spolu s modem snímků rozhodně nejlepší výsledky ze všech představených algoritmů. V porovnání s četnosti snímku si troufnu tvrdit, že má pro více snímků dokonce lepší výsledky. Navíc není tolik závislý na vnějších vlivech jako je osvětlení a všechny jeho parametry se dají nastavit celkem obecně pro většinu scén. U modu byl právě problém s nastavením prahu T na porovnání dvou intenzit pixelů na shodu. Je také výpočetně méně náročný než modus. Další výhodou je, že oproti ostatním řešením je model popsán matematicky daleko přesněji, a proto je na něj možno lépe aplikovat další vylepšení, které budou představeny v následující části. Na druhou stranu se opět jedná o jakýsi statistický přístup nad sesbíranými daty, a proto se i zde potýkáme s neduhy vznikajícími při výskytu pravděpodobnostně nedostatečné informace. Jak dopadl výsledek Gaussových funkcí ukazuje obrázek 4.6.

3.4.1 Implementace MOG

Velikou úsporou času byla pro mě tolik příjemná skutečnost, že samotný algoritmus je již implementovaný v knihovně OpenCv2.0. Zmíněná knihovna pro počítačové vidění je uvolněna jako open source, a tak jsem z ní mohl převzít kód týkající se MOG a upravit ho pro svou potřebu. Zároveň jsem se snažil, aby bylo možno vzniklý kód znovu začlenit, a tak je má funkce kompatibilní s interním rozhraním pro práci s modelem pozadí. V OpenCv je právě postupů sloužících k popsání pozadí implementováno více a všechny je možno volat stejnou funkcí a pouze rozdílný typ struktury *CvBGStatModel* popisující pozadí určí, která funkce bude volána.

Menší nevýhodou je, že algoritmus vyžaduje oproti ostatním představeným postupům nastavení více parametrů. V mém případě modeluje každý pixel pomocí 3 Gaussových funkcí, což je v souladu s literaturou, kde autoři ve své práci [12] navrhli jako rozumný počet distribucí v rozmezí 3–5. Rovněž práh příslušnosti hodnoty pixelu k jedné z distribucí jsem ponechal jako 2,5 násobek standardní odchylky σ. Naproti tomu hodnotu samotné standardní odchylky jsem volil menší, než je doporučeno pro video. U videa se předpokládá velké množství snímků a s každým dalším příchozím snímkem zapadajícím do nějakého rozložení se σ tohoto rozložení zmenšuje. To znamená, že se funkce více zužuje – specializuje na konkrétnější hodnoty. U jednotlivých snímků takový luxus postrádáme,

takže je nutné již funkce inicializovat menší standardní odchylkou za cenu větší náchylnosti vůči změně osvětlení. Čím menší hodnoty standardní odchylky, tím vznikne více distribucí popisující určitý úsek intenzity a jako důsledek nemusí být dva podobné pixely modelovány stejnou funkcí. Experimenty ukázaly, že hodnota optima se pohybuje kolem $\sigma <= 20^2$. Algoritmus pracuje s RGB modelem za předpokladu totožné standardní odchylky pro všechny tři složky vektoru. Parametr α nastavuji v závislosti na vstupním počtu snímků jako $\alpha = pocetSnimku^{-1}$. To znamená, že velikost klouzavého okna se rovná počtu snímků L a k aktualizaci modelu jsou využívány pouze první rovnice 2.38. Větší nebo menší velikost okénka dává jen vzniknout pouze dalšímu šumu a i ze své podstaty nemá smysl.

Ač se to na první pohled nemusí zdát, je algoritmus celkem náročný na paměť počítače. U videa se předpokládá malé rozlišení, takže to prakticky nehraje roli, ale při práci s fotografiemi můžeme narazit na nedostatek paměti. V originálním řešení je každý pixel fotografie popsán směsi Gaussových funkcí, tzn. pro každý pixel je alokovaná struktura obsahující sumu výskytů (`integer`), váhu (`double`), odchylku pro každý kanál (`double pole[3]`) a střední hodnotu pro každý kanál (`double pole[3]`), tj. $4 + 8 + 3 * 8 + 3 * 8 = 60$ bajtů (v závislosti na systému), které se musí alokovat na jednu Gaussovu distribuci. Uvažujeme-li 10 M pixelovou fotografii o rozměrech 3888 x 2592 se třemi distribucemi, pak je nutno alokovat minimálně $3888 * 2592 * 3 * 60 = 1\ 813\ 985\ 280$ bajtů paměti. Já jsem si z úsporných důvodů dovolil změnit strukturu tak, že jsem všechny čísla v plovoucí řadové čárce typu `double` nahradil typem `float`, který zabírá poloviční paměť a sumu výskytů jsem nahradil za bezznaménkovým `char` (1B), a tak snížil paměťové nároky na více jak polovinu. To vše bez ztráty na kvalitě algoritmu. Tuto úpravu jsem mohl udělat, protože počet příchozích fotografií bude v řádů desítek neboli počet « 255.

3.4.2 Detekce stínu a barevný model

Jelikož celý výpočet je prováděn v RGB prostoru, bylo nutné najít barevný model, který bude popisovat stíny také v RGB. Navíc by bylo výhodné, abychom mohli využít již některé spočítané hodnoty z předešlého algoritmu MOG. V článku [9] se rozhodli zvolit model z [8], kvůli požadovaným kritériím. Model se skládá s intenzitní a barevné složky. Pokud jsou hodnoty jak intenzitní, tak barevné složky aktuálního pixelu v mezích dané prahem, pak je pixel považován jako stín.

Implementace byla provedena podle teorie z kapitoly 2.9.3, přesto se ovšem neobešla bez problémů. Vše jsem zdárně vyřešil, až po nalezení diskuze na toto téma v [2], kde uvádějí, že se autoři dopustili malého překlepu. V článku [9] v kapitole 2.3 tvrdí, že α je v rozmezí 2,5 násobku standardní odchylky a $\tau < c < 1$. Správně má být c (v mém značení CD) v rozmezí 2,5 násobku standardní odchylky a $\tau < \alpha < 1$. Znamená to, že stín nalezneme vynásobením očekávané hodnoty E_i parametrem α, který je v uvedeném rozmezí. Tímto výrokem říkáme, že barvu zastíněného objektu očekáváme tmavší, než je jeho původní barva. Literatura uvádí jako vhodnou hodnotu parametru $\tau = 0,7$. Dále s předpokladem, že standardní odchylka ve všech dimenzích RGB je stejná můžeme zjednodušit výpočet α z rovnice 2.44 na:

$$\alpha = \frac{E_i I_i}{E_i^2} \tag{3.5}$$

kde pro počítaný pixel i je E_i RGB vektor označující očekávanou hodnotu pixelu neboli v našem případě střední hodnota Gaussova rozdělení μ. Proměnná I_i reprezentuje

RGB vektor pixelu na aktuálně zpracovávané fotografii. Výpočet je podobný rovnici (5) z článku [8].

Myšlenka vylepšení pozadí je taková, že hodnoty pixelů se stínem nahradíme hodnotami bez stínů. Samotné odstranění stínu se děje ve dvou krocích. Je to dáno povahou per-pixelových operací. V prvním kroku se nejprve hodnota každého příchozího pixelu porovná s hodnotou pozadí a určí se, zda se jedná o stín. Po zpracování všech pixelů v aktuální fotografii provedeme odstranění šumu (osamocených pixelů označených jako stín). V druhém kroku nahradíme hodnotu u zbylých pixelů obsahujících stín hodnotou pozadí.

Obrázek 3.3: Detekce a odstranění stínu. Vlevo originální snímek. Uprostřed extrakce pozadí. Vpravo odstranění stínu $\tau = 0,6$.

Detekce stínu při rozlišování popředí funguje celkem obstojně a implementovaná funkce jako taková by se jistě dala využít. Ostatně odstraňování stínů je neustále dokola se vyskytující úloha při extrakci pozadí. Nicméně výsledky aplikace při rekonstrukci byly zcela neuspokojivé. Ve skutečnosti vržené stíny mají na výsledné zrekonstruované pozadí pramalý vliv. Některé části obrazu, zejména ty zastíněné na delší dobu, vykazovaly nepatrné vizuální zlepšení, na druhou stranu jiné části obrazu se spíše zhoršily. Pro čtenářovu představu se zde bavíme o desítkách pixelů ve fotografiích čítajících sto tisíce pixelů. Zde ovšem musím uvést, že z důvodu úspory výpočetního výkonu byla prováděna detekce stínů současně s počítáním pozadí. To znamená, že stíny byly posuzovány vzhledem k dosud známému pozadí a ne vzhledem k celkovému modelu. Vysvětluje to ten fakt, že některé části obrazu se zlepšily, zatímco jiné zhoršily. Důvodem, proč je pracování se stíny nevhodné u rekonstrukce pozadí z několika fotografií, je fakt, že ve snímcích, ve kterých se mění celková intenzita osvětlení, bude jas odhadnutého pozadí intenzitně někde uprostřed, a to způsobí, že u snímků s větším výkyvem jasu vzhledem k modelu nebude detekce stínů fungovat správně pro zvolený práh τ. U videa je tento problém zanedbatelný, protože se jednoduše počká několik sekund, než se systém na změnu adaptuje, a tudíž může být práh τ stále zvolený napevno.

3.4.3 Vylepšení pomocí okolí pixelů

Silný šum ve výsledku mě vedl k úvahám, jak výstup vylepšit. Prvním krokem je vůbec detekování potenciálních chybných oblastí (se šumem). K problémům dochází v momentě, kdy jsou *fitness* hodnoty Gaussových funkcí velmi podobné. V originálním algoritmu se tento problém vůbec neřeší, protože jeho primárním účelem je detekce popředí. Jednoduše se za pozadí prohlásí prvních B Gaussových funkcí (tzv. multimodální mozadí). Kolik funkcí to bude ovlivňuje parametr T, viz vzorec 2.37. V mém případě musím vybrat za pozadí pouze jednu distribuci a smícháním více distribucí by se problém taktéž nevyřešil a rovněž

by vznikal šum. Za Gaussovu funkci, která správně modeluje pozadí označuji takovou funkci, která je zařazena na prvním místě a od druhé nejlepší distribuce se liší o hodnotu *fitness* větší než $0,1$. Ostatní funkce jsou označeny jako nerozhodnutelné, respektive potenciálně chybné (viz obr. 3.5).

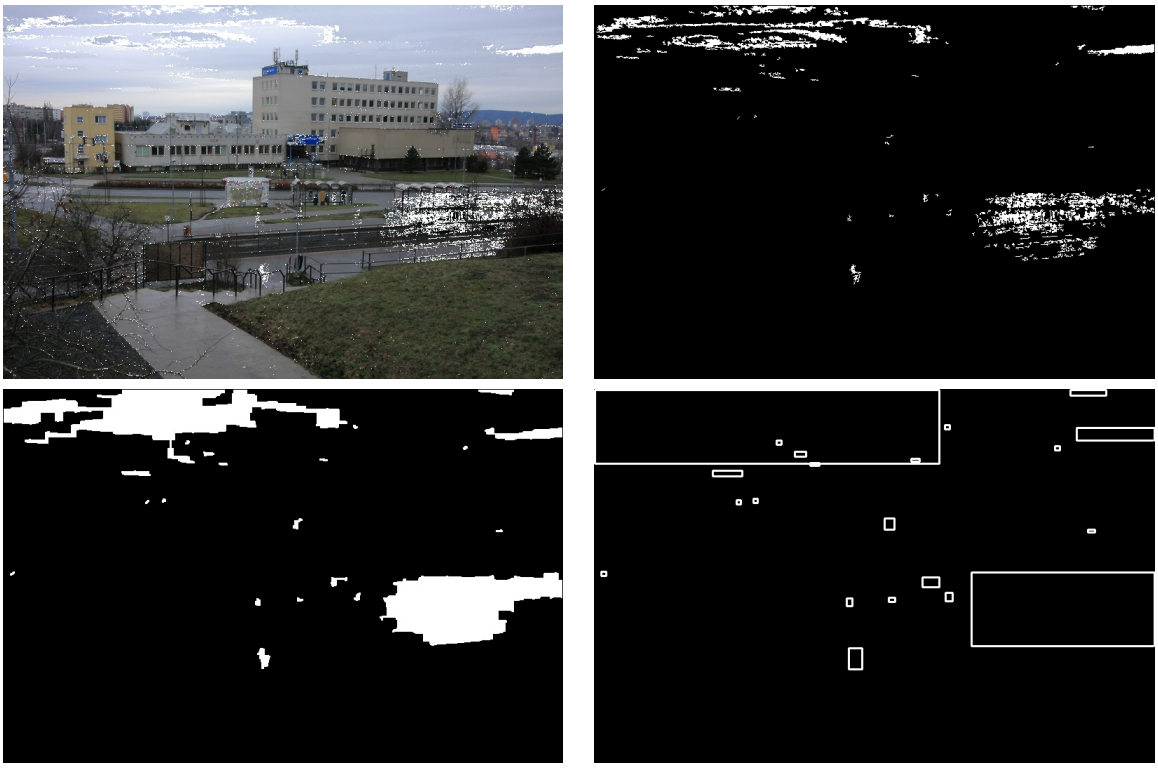
Ale zpátky k vylepšení. V první řadě jsem si zkusil odpovědět na otázku, zdali je možno chybný obraz nějak iterativně zlepšit za pomocí okolních pixelů. Je totiž pravděpodobné, že se hodnota pixelu pozadí nebude ve svém v okolí moc odlišovat. V každé iteraci budeme postupně procházet pozice potenciálně chybných pixelů P a srovnávat jejich hodnoty s 8–okolím. U pixelů P potřebujeme rozhodnout, kterou Gaussovu distribuci zvolit. Předpokládá se, že správná hodnota pozadí je v modelu MOG obsažena jako jedna s distribucí. Za finální distribuci modelující pozadí pixelu $P\{p_{xy}\}$ vybereme takovou, která se nejvíce vyskytuje v okolních pixelech. Má smysl pouze uvažovat takové okolní pixely, o nichž víme, že jsou správné (nejsou obsaženy v P). Úspěšně zpracovaný pixel je v další iteraci algoritmu z P vyjmut a může sloužit jako zdroj podle, kterého se počítá okolí.

Výsledky prokázaly, že je tento postup schopný vylepšit celkový obraz, ale k mému zklamání je vylepšení spíše kosmetické (podobně jak tomu bylo u kompenzace stínu). Algoritmus spíš funguje podobně jako filtr k vyhlazení jemného výstřelového šumu. Výsledek se vylepší jen u malých oblastí v řádu pixelů. Jakmile tvoří chybné oblasti pixelů P větší celky, program si s nimi již neporadí. Ukázka výstupu před a po „vylepšení“ je znázorněna na obrázku 3.4. Výstup je počítán pouze z 10 fotek, aby v něm vzniklo více chybných pixelů a efekt byl patrnější. U 36 fotografií není výsledek vizuálně odlišný prakticky vůbec.

Obrázek 3.4: Sotva postřehnutelný rozdíl při uvažování okolních pixelů: Vlevo původní výsledek. Vpravo výsledek po vylepšení za pomocí okolních pixelů (10 vstupních snímků). V pravé části obrazu nebylo skutečné pozadí vidět ani jednou a vyskytovaly se zde celou dobu jen tramvaje.

3.4.4 Vylepšení per region

Jak je vidět na obrázku 4.6, převážná část výsledného obrazu je celkem uspokojivá, nicméně oblasti s častým střídáním pozadí se projeví jako šum a působí rušivě. Toto je dáno povahou všech dosud uvedených algoritmů. Všechny totiž fungují na principu per–pixel, tj. zpracovávání obrazu po pixelech. Každý pixel je počítán samostatně, bez ohledu na jeho celkové okolí. Ukázalo se, že hlavním problémem všech algoritmů jsou částečné stacionární objekty. Konkrétně časté přijíždění a odjíždění tramvají. Osobně považuji za uspokojivé řešení takové, které dá uživateli čistý výstup. Proto je myslím zcela v pořádku prohlásit za součást pozadí objekt (tramvaj), pokud setrval na svém místě dostatečně dlouho. Samozřejmě úplné vyčištění scény i přes to, že by se skutečné pozadí vyskytovalo ve snímcích třeba jen jednou či dvakrát, je úkol pro počítač nadmíru obtížný a k jeho vyřešení byla by potřeba sofistikovaná umělá inteligence. Naopak pro člověka toto není žádný problém.



Obrázek 3.5: Vlevo nahoře výsledek MOG s potenciálními chybnými pixely označenými bíle. Vpravo nahoře binární obraz s těmito oblastmi bez osamocených pixelů. Vlevo dole uzavření nad binární množinou potenciálních chybných pixelů. Vpravo dole orámování spojených oblastí (36 vstupních snímků).

Zaručení vizuální spojitosti objektu bez přítomnosti šumu lze provést pouze operacemi počítanými nad většími oblastmi, než je jeden pixel, tzv. per–region operacemi. Jednu takovou operaci jsem ve svém řešení navrhnul. Cílem je ohraničit potenciální chybné oblasti ve snímku obdélníkovým výřezem a nahradit celý výřez tím nejpravděpodobnějším v rámci sady snímků. Práce s celými výřezy tvaru obdélníka je výhodná, protože se s nimi v zásadě lépe pracuje a navíc lze využít optimalizovanou funkci z knihovny OpenCV *cvMatchTemplate*. Na obrázku 3.5 jsou vpravo nahoře ilustrovány chybné pixely tvořící vizuální shluky, které ovšem nejsou spojité. Ze všeho nejprve musíme oblasti, které k sobě vizuálně patří sloučit dohromady, abychom je mohli detekovat a orámovat. Provedl jsem to morfologickou operací uzavření (viz podkapitola 2.3), která slije blízké objekty dohromady a vyplní díry ve výsledném objektu. Operaci uzavření realizuji tak, že nad obrazem několikrát za sebou provedu dilataci následovanou totožným počtem erozí se stejným strukturním elementem 3 x 3 pixely. Počet iterací morfologických operací je závislý na rozlišení obrazu. Ještě před uzavřením není na škodu celý binární obraz nepatrně zvětšit funkcí dilatace. Vznikne binární maska M využívaná později. Výše uvedeným postupem dostaneme celistvé bílé oblasti, které jsem již schopen algoritmem „connected component“ sloučit dohromady a považovat za jeden objekt. Vzniknuvší objekty orámuji nejmenší možnou nerotovanou čtvercovou obálkou. Obálky reprezentují výřezy ve fotografiích. Příslušné výřezy budu mezi sebou přes všechny snímky porovnávat. Postup orámování je znázorněn na obrázku 3.5.

Jednotlivé odpovídající si výřezy ze všech fotografií jsou mezi sebou porovnávány a na

základě podobnosti zařazeny do tříd. Porovnání na podobnost předchází předzpracování výřezů. Regiony jsou převedeny na šedotónový obraz v rozsahu 0–255 hodnot a vyhlazeny Gaussovým konvolučním filtrem 3 x 3 pixely. Filtr obraz rozmaže a bude tak méně náchylný na šum. Překvapivě se příliš neosvědčila ekvalizace histogramu, takže jsem od ní nakonec upustil. Na výřezy se dívám jako na 2D signály, které mezi sebou porovnávám normalizovanou cross-korelací, viz kapitola 2.5. Výsledkem korelační funkce je reálné číslo v rozmezí $< -1, 1 >$ nazývané korelační koeficient. Korelační koeficient udává podobnost mezi dvěma signály. Čím větší číslo, tím jsou si signály podobnější. Koeficient 1 je shoda. Za dva vzájemně podobné výřezy pokládám signály s korelačním koeficientem $NCCF > 0,945$. K této hodnotě jsem dospěl experimentálně. Porovnáním výřezů mezi sebou vznikne n kategorií se vzájemně si podobnými členy. Finální výřez vyberu z kategorie s největším počtem členů. Výřez lze zvolit náhodně, ale já jsem se snažil opět vybrat ten nejvhodnější. Obálka ohraničující potenciálně problémové pixely je větší než objekt v ní obsažený, a tím pádem obsahuje i správné pixely a je výhodné tyto pixely využít ke zpřesnění. Nad vybranou kategorií spustím ještě jedno porovnání korelací. Budu hledat takový výřez, který je nejvíce podobný (jeho $NCCF$ je maximální) původnímu výsledku známému ze směsi Gaussových funkcí. Porovnání se ovšem provede pouze nad pixely neodpovídající binární masce M, tj. pixely, o kterých algoritmus usoudil, že jsou správné. Testování výkonnosti algoritmu ukázalo, že cílené zvolení výřezu v rámci kategorie zvyšuje pravděpodobnost selekce vhodnějšího pozadí. Na závěr je vhodné vyrovnat jas všech výřezů doplněných do snímku pozadí. Jas jsem se zejména kvůli jednoduchosti implementace rozhodl vyrovnat lineárně. Samotné vyrovnání jsem nerozlišoval na snižování nebo zvyšování jak je to v sekci 2.6, ale použil klasickou rovnici přímky $I'(x, y) = kI(x, y)$ s hlídám přetečení.

Vylepšení per-region z obrazu odstraní všechen šum. Výsledný obraz působí čistě a nerušivě, byť může obsahovat některé objekty popředí. Je to však lepší než přítomnost šumu v tomto místě. V zásadě shledávám výsledky jako velmi dobré. Nevýhodou je vyšší výpočetní náročnost korelace. Výsledek vylepšení per-region je patrný na obrázku 4.7.

Kapitola 4

Výsledky

V této kapitole jsou především zobrazeny výsledné fotografie všech probraných metod. Algoritmy byly testované na dvou sadách fotografií. První sadou byla nám dobře známá zastávka hromadné dopravy a její specifika jsou diskutovány na začátku kapitoly 3. Druhá sada je podstatně menší a zobrazuje ulici poblíž centra města. Plné rozlišení fotografií bylo 3888 x 2592 pixelů. Je doporučeno spouštět program s minimálně 2 GB RAM. Obě kompletní sady si je možné prohlédnout v přílohách.

Druhá sada snímků obsahuje nekomplikované pozadí a byla použita na testování výkonnosti algoritmů na malém vzorku dat. Nevýhodou směsi Gaussových funkcí – MOG je delší počáteční adaptace. U méně snímků (počet < 10) dává většinou horší výsledky než modus. U méně fotografií, kde měl bezchybné výsledky i medián, se u nevylepšeného MOG objevovaly artefakty. Rozdíl je vidět na obrázcích 1 a 2. V takovémto případě je častokrát výhodnější použít četnost snímků. Naopak u více fotografií je nevylepšený MOG nejlepší, navíc modus je na to kvůli své výpočetní složitosti nevhodný. MOG každou fotografii zpracovává konstantní čas. Vylepšení MOG za pomocí regionů si vedlo ze všech algoritmů nejlépe a poradilo si vůbec s nejmenším počtem snímků, viz obr. 3.

Obrázek 4.1: Průměr z 36 vstupních snímků

Obrázek 4.2: Medián z 36 vstupních snímků

Obrázek 4.3: Modus HSV obrazu – v potaz brány pouze 2 kanály H a S (36 vstupních snímků)

Obrázek 4.4: Modus RGB obrazu (36 vstupních snímků)

Obrázek 4.5: Modus šedotónového obrazu (36 vstupních snímků)

Obrázek 4.6: Nevylepšený mixture of Gaussian. Vylepšení za pomocí stínů vypadá velmi podobně (36 vstupních snímků).

Obrázek 4.7: Vylepšený mixture of Gaussian použitím per-region operací (36 vstupních snímků).

Kapitola 5

Závěr

V diplomové práci jsem se zabýval jak základními, tak i pokročilejšími metodami extrakce pozadí z videa a snažil se je aplikovat na fotografie. Mým cílem bylo nejenom tyto metody pochopit a implementovat, ale především je jakýmkoli způsobem vylepšit. Každou metodu jsem se snažil co nejvíce rozebrat a srovnat její výkonnost s předešlými. Všechny problémy a postupy, které jsem použil, jsou popsány v teoretické části.

Zdaleka nejhůře si vedlo průměrování snímků. Čistý výstup tato metoda nedala ani v jednom případě. K dosažení dobrých výsledků by ji bylo nutné aplikovat na velký počet fotografií s nekomplikovaným pozadím. Jediný primát si průměrování drží v rychlosti. Druhou metodou byl medián snímků. Medián se již dá na nekomplikované pozadí s úspěchem použít. Na základě teoretických poznatků jsem mohl medián počítat v lineárním čase. Třetí metoda, modus snímků, přinesla ještě lepší výsledky než medián. Bohužel její výpočet roste s počtem snímků kvadratickou složitostí. Právě z důvodů časové složitosti jsem zkoumal využití barevných prostorů kvůli snížení počtu kanálů. Velmi podobných výsledků jako u RGB obrazu jsem dosáhl při experimentech s šedotónovým obrazem. Tímto se výpočet urychlil 3x bez znatelné ztráty kvality. Obecně platí, že lepší výsledky než průměr bude mít vždy medián a lepší než medián zase modus. Stejné pořadí platí i pro jejich rychlost. Poslední zkoumanou metodou, na které bylo aplikováno zároveň nejvíce vylepšení, byla směs Gaussových funkcí – MOG. Metoda statisticky popisuje soubor dat daleko dokonaleji než předchozí metody, a proto se také lépe pracovalo na dalších vylepšeních. Naprosto nevyhovující korekcí se ukázalo vylepšení pomocí okolních pixelů i pomocí stínů. Naopak za velmi dobrý krok považuji nápad zlepšit výsledek za pomocí regionů. Výsledek byl zcela bez šumu a působil velmi dobrým dojmem. Myslím, že vylepšení per-region jako takové by se dalo použít i u jiných metod. Nevýhodou MOG je delší počáteční adaptace a s tím spjaté horší výsledky u méně snímků. Naopak u více fotografií dává nevylepšený MOG nejlepší výsledky. Vylepšení MOG za pomocí regionů zaznamenalo úspěch i u méně fotografií a poskytlo lepší výsledky než modus. Osobně ji tedy považuji za univerzální metodu.

V rámci další budoucí práce by bylo možno pracovat s fotografiemi focenými „z ruky“ (bez použití stativu) a jemně tak navázat na mou bakalářskou práci *Panoramatické snímky automaticky*. To by obnášelo najít ve fotografiích význačné body, tyto body vzájemně spárovat a najít vhodné transformace tak, aby fotografie na sobě seděly stejně, jako kdyby byly pořízeny ze stativu. Autor si myslí, že taková vlastnost programu by byla uživatelem velmi vítaná, protože odpadne práce ze stativem. Na druhou stranu by si musel být fotograf vědom toho, že může za své pohodlí zaplatit výsledným menším výřezem z fotografií způsobeným výchylkami fotoaparátu (klepání rukou). Dá se taktéž předpokládat, že nutné transformace snímků mohou vést ke zhoršení kvality algoritmu.

Literatura

[1] Open Source Computer Vision Library – OpenCV.
URL <http://www.intel.com/technology/computing/opencv/index.htm>

[2] Yahoo group – OpenCV.
URL <http://tech.groups.yahoo.com/group/OpenCV>

[3] Image processing learning resources. [online], [cit. 2010-03-26].
URL <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm>

[4] Selection (deterministic & randomized): finding median in linear time. [online], [cit. 2010-05-11].
URL <http://www.cs.cmu.edu/afs/cs/academic/class/15451-s09/www/lectures/lect0122.pdf>

[5] Bruchanov, M.: Základy zpracování obrazů.
URL <http://bruxy.regnet.cz>

[6] Haritaoglu, I.; Harwood, D.; Davis, L. S.: W^4: Who? when? where? what? a real time system for detecting and tracking people. In *Third Face and Gesture Recognition Conference*, 1998, s. 222–227.

[7] Hlaváč, V.: Jasové a geometrické transformace. [online], [cit. 2010-04-27].
URL <http://cmp.felk.cvut.cz/~hlavac>

[8] Horprasert, T.; Harwood, D.; Davis, L. S.: A statistical aproach for real-time robust background subtraction and shadow detection. Technická zpráva, Computer vision laboratory University of Maryland, College Park.

[9] KaewTraKulPond, P.; Bowden, R.: An improved adaptive background mixture models for real-time tracking with shadow detection. In *2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[10] McIvor, A.: Background substraction techniques. Technická zpráva, Remuera, Auckland, New Zeland.

[11] Pavlidis, I.; Morellas, V.; Tsiamyrtzis, P.; aj.: Urban surveillance systems: from the laboratory to the commercial world. In *Proceedings of the IEEE vol. 89, no. 10*, 2001, s. 1478–1497.

[12] Stauffer, C.; Grimson, W.: Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999.

[13] Stockham, T. G.: Image processing in the context of a visual model. In *Proceedings of the IEEE vol. 60, no. 7*, 1972, s. 828–842.

[14] Tamerson, B.: Background substraction. [online], 29.09.2009, naposledy navštíveno 2009-12-20.
URL <http://www.cs.utexas.edu/~grauman/courses/fall2009/slides/lecture9_background.pdf>

[15] Wikipedia: HSV. [online], [cit. 2009-12-17].
URL <http://cs.wikipedia.org/wiki/HSV>

[16] Wikipedia: RGB. [online], [cit. 2009-12-17].
URL <http://cs.wikipedia.org/wiki/RGB>

[17] Wikipedia: Taxicab geometry. [online], [cit. 2009-12-18].
URL <http://en.wikipedia.org/wiki/Taxicab_geometry>

[18] Wikipedia: Medián. [online], [cit. 2010-01-01].
URL <http://cs.wikipedia.org/wiki/Medi%C3%A1n>

[19] Wikipedia: Modus. [online], [cit. 2010-01-01].
URL <http://cs.wikipedia.org/wiki/Modus>

[20] Wikipedia: Optical flow. [online], [cit. 2010-01-01].
URL <http://en.wikipedia.org/wiki/Optical_flow>

[21] Wikipedia: Connected component labeling. [online], [cit. 2010-03-26].
URL <http://en.wikipedia.org/wiki/Connected_Component_Labeling>

[22] Wikipedia: Mathematical morphology. [online], [cit. 2010-03-26].
URL <http://en.wikipedia.org/wiki/Morphological_image_processing>

[23] Wikipedia: Gamma correction. [online], [cit. 2010-04-27].
URL <http://cs.wikipedia.org/wiki/Gamma_correction>

[24] Wikipedia: Histogram equalization. [online], [cit. 2010-04-28].
URL <http://en.wikipedia.org/wiki/Histogram_equalization>

[25] Wikipedia: Selection algorithm. [online], [cit. 2010-05-11].
URL <http://en.wikipedia.org/wiki/Selection_algorithm>

[26] Xiao, M.; Han, C.; Kang, X.: A background reconstrucition for dynamic scenes. Technická zpráva, Jiaotong University, Shaanxi P.R.China.

[27] Španěl, M.; Beran, V.: Přednášky FIT pro předmět Zpracování obrazu ZPO (Matematická morfologie). [online], [cit. 2010-04-27], dostupnost na privátních stránkách předmětu ZPO.

[28] Černocký, J.: Zpracování řečových signálů – studijní opora FIT. [online], 6.12.2006, [cit. 2010-03-23], dostupnost na privátních stránkách předmětu ZRE.

Seznam příloh

1. Základní manuál programu
2. CD/DVD
3. Testovací data

1. Základní manuál programu

Vstupem programu je soubor barevných fotografií stejných rozměrů, výstupem jediná fotografie totožných vlastností s odstraněným popředím. Jedná se o konzolovou aplikaci umožnující dávkové zpracování. Proto se oproti původní verzi nezobrazuje uživateli výsledek na monitoru a jméno výstupní fotografie je do jisté míry unikátní. Výstupní fotografie je uložena do kořenového adresáře aplikace a má jméno složené z názvu metody a číselného rozmezí zadaného jako poslední parametr. Program se spouští se 3 pevnými parametry v předem definovaném pořadí a jedním volitelným parametrem na konci. Pokud je některý parametr chybný nebo nesmyslný nebo se nezdaří načtení některé vstupní fotografie z kořenového adresáře, je tato skutečnost oznámena uživateli a program se ukončí.

Formát parametrů:

```
dip.exe -mean|-median|-mode|-mog imageMask firtNum-lastNum [-r|-ch]
```

Význam jednotlivých parametrů:

- `-mean | -median | -mode | -mog` – zvolí metodu výpočtu pozadí;
  - `-mean` – průměr snímků;
  - `-median` – medián snímků;
  - `-mode` – modus snímků počítaný na šedotónovém obrazu;
  - `-mode ... -ch` – modus snímků počítaný pro každý kanál zvlášť;
  - `-mog` – mixture of Gaussian;
  - `-mog ... -r` – mixture of Gaussian s vylepšením per-region.
- `imageMask` – Maska sloužící k zadávání vstupních fotografií. Je popsána regulárním výrazem `.*?+.*`. Protože se však podle ní budou načítat existující soubory, musí dále respektovat správný tvar zápisu souboru včetně přípony označující jeho typ. Za otazník si program nahradí čísla v rozmezí od `firsNum` do `lastNum`, a tak zjistí, které fotografie má načíst. Počet otazníků reprezentuje počet číslic. Např. maska `?.jpg 1-5` znamená soubory *1.jpg*, *2.jpg*, ..., kdežto maska `???.jpg 1-5` soubory *001.jpg*, *002.jpg*, ...
- `firstNumb-lastNumb` – `firsNum` je první číslo, které se postupně inkrementuje a nahrazuje se jím maska, dokud se nenarazí na `lastNumb`. Náhrada je včetně obou čísel.

Příklad spuštění:

```
dip.exe -mean img??.jpg 55-69
```

Při této konfiguraci program načte 15 obrázků s názvy *img55.jpg* až *img69.jpg*, ty zprůměruje a výsledek uloží do souboru *mean_img55–69.jpg*.

2. CD/DVD

Nosič má následující strukturu:

/images	testovací obrázky a spustitelný program
/source	zdrojové soubory programu
/text	zdrojové soubory pro LaTeX
readme.txt	informace k projektu

3. Testovací data

Výsledky dosažené na druhé sadě fotografií a ukázka obou testovacích sad.

Obrázek 1: Medián (10 vstupních snímků)

Obrázek 2: Nevylepšený MOG má problémy u méně snímků (10 vstupních snímků).

Obrázek 3: Vlevo modus, vpravo MOG s vylepšením per-region (4 vstupní snímky)

Obrázek 4: Druhá vstupní sekvence 10 snímků

Obrázek 5: První vstupní sekvence 36 snímků